Altova RaptorXML Server 2015

Benutzer- und Referenzhandbuch

Altova RaptorXML Server 2015 Benutzer- und Referenzhandbuch

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2014

© 2014 Altova GmbH

Inhaltsverzeichnis

1 Informationen zu RaptorXML Server 3

1.1 Editionen und Schnittstellen 5
1.2 Systemanforderungen 7
1.3 Funktionalitäten 8
1.4 Unterstützte Spezifikationen 10

2 Einrichten von RaptorXML 12

2.1 Einrichten unter Windows 13
2.1.1 Installation unter Windows 14
2.1.2 Lizenzierung unter Windows 17
2.2 Einrichten unter Linux 21
2.2.1 Installation unter Linux 22
2.2.2 Lizenzierung unter Linux 25
2.3 Einrichten auf Mac OS X 27
2.3.1 Installation auf Mac OS X 28
2.3.2 Lizenzierung auf Mac OS X 31
2.4 XML-Kataloge 33
2.4.1 Funktionsweise von Katalogen 34
2.4.2 Der XML-Katalogmechanismus von Altova 36
2.4.3 Variablen für Windows-Systempfade 39
2.5 Globale Ressourcen 41
2.6 Sicherheitsfragen 43

3 Befehlszeilenschnittstelle (CLI) 46

3.1 XML-, DTD-, XSD-Validierungsbefehle 48
3.1.1 valxml-withdtd (xml) 49
3.1.2 valxml-withxsd (xsi) 53
3.1.3 valdtd (dtd) 59
3.1.4 valxsd (xsd) 62
3.1.5 valany 67
3.2 Befehle für die Überprüfung der Wohlgeformtheit 71
3.2.1 wfxml 72

3.2.2 wfdtd ... 76
3.2.3 wfany ... 79
3.3 XSLT-Befehle ... 82
3.3.1 xslt ... 83
3.3.2 valxslt ... 90
3.4 XQuery-Befehle ... 96
3.4.1 xquery ... 97
3.4.2 valxquery ... 102
3.4.3 valxqueryupdate ... 106
3.5 Die Befehle zur Hilfe und Lizenzierung ... 110
3.5.1 Hilfebefehl ... 111
3.5.2 Lizenzbefehle ... 113
3.6 Lokalisierungsbefehle ... 114
3.6.1 exportresourcestrings ... 115
3.6.2 setdeflang ... 116
3.7 Optionen ... 117
3.7.1 Kataloge, globale Ressourcen, ZIP-Dateien ... 118
3.7.2 Meldungen, Fehler, Hilfe, Timeout, Version ... 119
3.7.3 Verarbeitung ... 120
3.7.4 XML ... 121
3.7.5 XSD ... 122
3.7.6 XQuery ... 124
3.7.7 XSLT ... 127

4 HTTP-Schnittstelle 130

4.1 Einrichten des Servers ... 132
4.1.1 Starten des Servers ... 133
4.1.2 Testen der Verbindung ... 135
4.1.3 Konfigurieren des Servers ... 136
4.2 Client Requests ... 141
4.2.1 Initiieren von Aufträgen mittels POST ... 144
4.2.2 Server-Antwort auf den POST Request ... 149
4.2.3 Abrufen des Ergebnisdokuments ... 152
4.2.4 Abrufen von Fehler-/Meldungs-/Ausgabedokumenten ... 156
4.2.5 Freigeben von Serverressourcen nach der Verarbeitung ... 158

5 Python-Schnittstelle 160

5.1 Erstellen von Python-Skripts ... 162

5.2 Ausführen von Python-Skripts ... 165
5.3 Beispielskript 01: Verarbeiten von XML ... 166
5.3.1 Skriptcode ... 167
5.3.2 Ergebnisdokument ... 170
5.4 Beispielskript 02: XML neu formatieren ... 171
5.4.1 Skriptcode ... 172
5.4.2 Ergebnisdokument ... 175
5.5 Python-API: Das Auftragsobjekt ... 178
5.6 Python-XML-API ... 179
5.6.1 xml.Attribute ... 181
5.6.2 xml.Character ... 182
5.6.3 xml.Comment ... 183
5.6.4 xml.Document ... 184
5.6.5 xml.Namespace ... 185
5.6.6 xml.Element ... 186
5.6.7 xml.Notation ... 188
5.6.8 xml.NSAttribute ... 189
5.6.9 xml.ProcessingInstruction ... 190
5.6.10 xml.QName ... 191
5.6.11 xml.UnexpandedEntityReference ... 192
5.6.12 xml.UnparsedEntity ... 193
5.7 Python-XSD-API ... 194
5.7.1 xsd.Annotation ... 200
5.7.2 xsd.Any ... 201
5.7.3 xsd.AnyAttribute ... 202
5.7.4 xsd.Assertion ... 203
5.7.5 xsd.AttributeDeclaration ... 204
5.7.6 xsd.AttributeGroupDefinition ... 205
5.7.7 xsd.AttributePSVI ... 206
5.7.8 xsd.AttributeUse ... 208
5.7.9 xsd.Block ... 209
5.7.10 xsd.ComplexTypeDefnition ... 210
5.7.11 xsd.ContentType ... 211
5.7.12 xsd.Defined ... 212
5.7.13 xsd.DerivationMethod ... 213
5.7.14 xsd.ENTITY ... 214
5.7.15 xsd.ElementDeclaration ... 215
5.7.16 xsd.ElementPSVI ... 217
5.7.17 xsd.Final ... 219
5.7.18 xsd.ID ... 220

5.7.19.... xsd.IDREF 221
5.7.20.... xsd.ID_IDREF_binding 222
5.7.21.... xsd.ID_IDREF_table 223
5.7.22.... xsd.IdentityConstraintDefinition 224
5.7.23.... xsd.Instance 226
5.7.24.... xsd.ModelGroup 227
5.7.25.... xsd.ModelGroupDefinition 228
5.7.26.... xsd.NCName 229
5.7.27.... xsd.NMTOKEN 230
5.7.28.... xsd.NOTATION 231
5.7.29.... xsd.Name 232
5.7.30.... xsd.NamespaceBinding 233
5.7.31.... xsd.NamespaceConstraint 234
5.7.32.... xsd.NotationDeclaration 235
5.7.33.... xsd.OpenContent 236
5.7.34.... xsd.PSVI 237
5.7.35.... xsd.Particle 238
5.7.36.... xsd.QName 239
5.7.37.... xsd.Schema 240
5.7.38.... xsd.Scope 242
5.7.39.... xsd.Sibling 243
5.7.40.... xsd.SimpleTypeDefinition 244
5.7.41.... xsd.TypeAlternative 246
5.7.42.... xsd.TypeTable 247
5.7.43.... xsd.Unbounded 248
5.7.44.... xsd.ValueConstraint 249
5.7.45.... xsd.XPathExpression 250
5.7.46.... Spezielle vordefinierte Datentypobjekte 251
5.7.47.... String-Datentypobjekte 252
5.7.48.... Boolesches Datentypobjekt 253
5.7.49.... Objekte vom Datentyp Number 254
5.7.50.... Objekte vom Datentyp Duration 255
5.7.51.... Objekte vom Datentyp Date und Time 256
5.7.52.... Objekte vom Datentyp Binary 257
5.7.53.... Facet-Objekte 258

6 Java-Schnittstelle 262

6.1..... Java-Beispielprojekt 264
6.2..... RaptorXML-Schnittstellen für Java 266
6.2.1..... RaptorXMLFactory 267

6.2.2 XMLValidator ... 274
6.2.3 XSLT ... 285
6.2.4 XQuery ... 293
6.2.5 RaptorXMLException ... 302

7 COM- und .NET-Schnittstelle 304

7.1 Informationen zur COM-Schnittstelle ... 305
7.2 Informationen zur .NET-Schnittstelle ... 306
7.3 Programmiersprachen ... 308
7.3.1 COM-Beispiel: VBScript ... 309
7.3.2 .NET-Beispiel: C# ... 312
7.3.3 .NET-Beispiel: Visual Basic .NET ... 315
7.4 API-Referenz ... 317
7.4.1 Schnittstellen ... 318
IServer ... 318
IXMLValidator ... 321
IXSLT ... 326
IXQuery ... 331
7.4.2 Enumerationen ... 338
ENUMAssessmentMode ... 338
ENUMErrorFormat ... 339
ENUMLoadSchemalocation ... 339
ENUMQueryVersion ... 340
ENUMSchemaImports ... 341
ENUMSchemaMapping ... 342
ENUMValidationType ... 342
ENUMWellformedCheckType ... 343
ENUMXMLValidationMode ... 344
ENUMXQueryVersion ... 345
ENUMXQueryUpdatedXML ... 345
ENUMXSDVersion ... 346
ENUMXSLTVersion ... 347

8 Informationen zum XSLT- und XQuery-Prozessor 350

8.1 XSLT 1.0 ... 351
8.2 XSLT 2.0 ... 352
8.3 XSLT 3.0 ... 355

8.4 XQuery 1.0 ... 356
8.5 XQuery 3.0 ... 360

9 XSLT- und XPath/XQuery-Funktionen 362

9.1 Altova-Erweiterungsfunktionen ... 364
9.1.1 XSLT-Funktionen ... 366
9.1.2 XPath/XQuery-Funktionen: Datum und Uhrzeit ... 369
9.1.3 XPath/XQuery-Funktionen: String ... 383
9.1.4 XPath/XQuery-Funktionen: Diverse Funktionen ... 389
9.1.5 Diagrammfunktionen ... 398
XML-Struktur von Diagrammdaten ... 402
Beispiel: Diagrammfunktionen ... 407
9.1.6 Barcode-Funktionen ... 411
9.2 Diverse Erweiterungsfunktionen ... 414
9.2.1 Java-Erweiterungsfunktionen ... 415
Benutzerdefinierte Klassendateien ... 416
Benutzerdefinierte Jar-Dateien ... 419
Java: Konstruktoren ... 420
Java: Statische Methoden und statische Felder ... 421
Java: Instanzmethoden und Instanzfelder ... 421
Datentypen: XPath/XQuery in Java ... 422
Datentypen: Java in XPath/XQuery ... 423
9.2.2 .NET-Erweiterungsfunktionen ... 425
.NET: Konstruktoren ... 427
.NET: Statische Methoden und statische Felder ... 428
.NET: Instanzmethoden und Instanzfelder ... 429
Datentypen: XPath/XQuery in .NET ... 430
Datentypen: .NET in XPath/XQuery ... 431
9.2.3 MSXSL-Skripts für XSLT ... 432

10 Altova LicenseServer 436

10.1 Informationen zum Netzwerk ... 438
10.2 Installation (Windows) ... 439
10.3 Installation (Linux) ... 440
10.4 Installation (Mac OS X) ... 442
10.5 Altova ServiceController ... 443
10.6 Zuweisen von Lizenzen ... 444
10.6.1 Starten von LicenseServer ... 445

10.6.2....Öffnen der LicenseServer-Konfigurationsseite (Windows) 447
10.6.3....Öffnen der LicenseServer-Konfigurationsseite (Linux) 450
10.6.4....Öffnen der LicenseServer-Konfigurationsseite (Mac OS X) 453
10.6.5....Laden von Lizenzen in LicenseServer 456
10.6.6....Registrieren von Produkten 460
Registrieren von FlowForce Server 460
Registrieren von MapForce Server 465
Registrieren von StyleVision Server 466
Registrieren von RaptorXML(+XBRL) Server 468
Registrieren von MobileTogether Server 470
10.6.7....Zuweisen von Lizenzen zu registrierten Produkten 471
10.7...Referenz zur Konfigurationsseite 477
10.7.1....License Pool 478
10.7.2....Server Management 483
10.7.3....Server Monitoring 488
10.7.4....Settings 489
10.7.5....Messages, Log Out 494

Index

Kapitel 1

Informationen zu RaptorXML Server

1 Informationen zu RaptorXML Server

Altova **RaptorXML Server** (in der Folge als RaptorXML bezeichnet) ist Altovas ultraschneller XML- und XBRL*-Prozessor der dritten Generation, der für die neuesten Standards und parallele Rechnerumgebungen optimiert wurde. RaptorXML lässt sich plattformübergreifend einsetzen und ermöglicht dank der Nutzung moderner Multi-Core Computer die ultraschnelle Verarbeitung von XML- und XBRL-Daten.

*** Anmerkung:** Die XBRL-Verarbeitung steht nur in RaptorXML+XBRL Server, nicht aber in RaptorXML Server zur Verfügung.

Editionen und Betriebssysteme
RaptorXML steht in Form von zwei verschiedenen Editionen zur Verfügung. Jede davon eignet sich für unterschiedliche Zwecke. Diese drei Editionen sind im Abschnitt Editionen und Schnittstellen beschrieben. RaptorXML steht für Windows, Linux und Mac OS X zur Verfügung. Nähere Informationen zu unterstützten Systemen finden Sie im Abschnitt Systemanforderungen.

Funktionalitäten und unterstützte Spezifikationen
RaptorXML unterstützt die Validierung von XML-Dateien, die XSLT-Transformation und XQuery-Ausführung. Für jede dieser Aufgaben stehen zahlreiche Optionen zur Verfügung. Im Abschnitt Funktionalitäten finden Sie eine umfangreiche Liste wichtiger Schlüsselfunktionen. Der Abschnitt Unterstützte Spezifikationen enthält eine detaillierte Liste der Spezifikationen, die RaptorXML erfüllt. Nähere Informationen finden Sie auf der RaptorXML-Seite der Altova-Website.

Diese Dokumentation
Diese Dokumentation ist im Lieferumfang der Applikation enthalten und steht auch auf der Altova Website zur Verfügung. Beachten Sie, dass sich im Browser Chrome Einträge des Inhaltsverzeichnisses nicht erweitern lassen, wenn die Dokumentation lokal geöffnet wird. Das Inhaltsverzeichnis in Chrome funktioniert korrekt, wenn die Dokumentation über einen Webserver geöffnet wird.

Diese Dokumentation ist in die folgenden Abschnitte gegliedert:

- Informationen zu RaptorXML (dieser Abschnitt)
- Einrichten von RaptorXML
- Befehlszeilenschnittstelle
- HTTP-Schnittstelle
- Python-Schnittstelle
- Java-Schnittstelle
- COM/.NET-Schnittstelle
- Informationen zum XSLT- und XQuery-Prozessor
- XSLT- und XPath/XQuery-Funktionen

- Altova LicenseServer

Letzte Aktualisierung: 15-09-2014

© 2014 Altova GmbH

1.1 Editionen und Schnittstellen

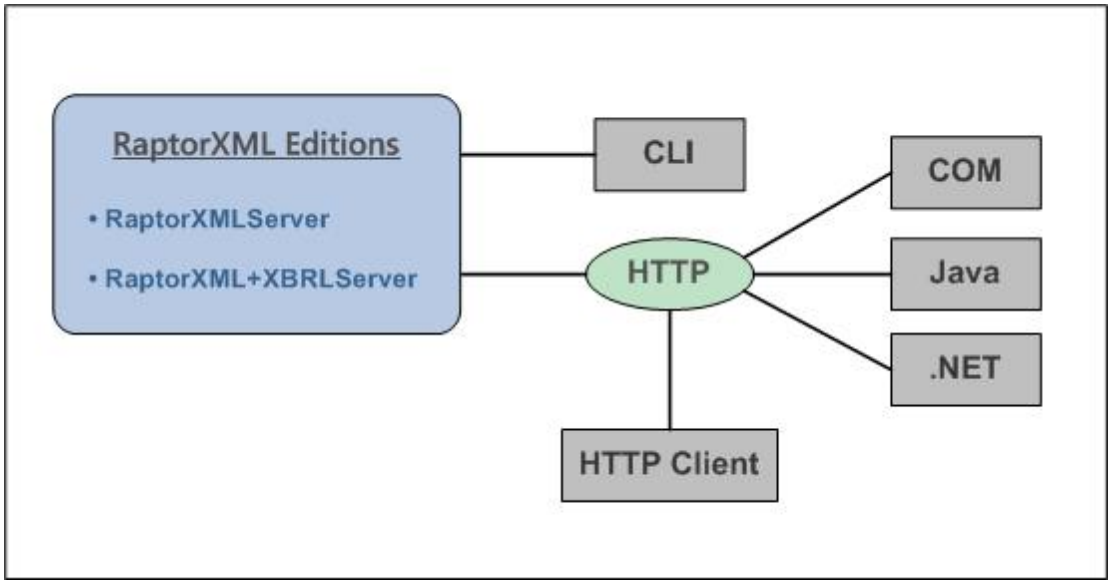
RaptorXML steht in Form der folgenden Editionen zur Verfügung:

- *RaptorXML Server* ist ein sehr schneller Prozessor zur Verarbeitung von XML, der XML, XML-Schema, XSLT, XPath, XQuery und mehr.
- *RaptorXML+XBRL Server* unterstützt alle Funktionen von RaptorXML Server und kann zusätzlich XBRL-Daten verarbeiten und validieren.

Schnittstellen
RaptorXML kann über die folgenden Schnittstellen aufgerufen werden:

- eine Befehlszeilenschnittstelle (CLI)
- eine COM-Schnittstelle auf Windows-Systemen
- eine .NET-Schnittstelle auf Windows-Systemen
- eine Java-Schnittstelle auf Windows-, Linux- und MacOS-Systemen
- eine HTTP-Schnittstelle, auf die von einem HTTP-Client aus zugegriffen werden kann
- eine Python-Schnittstelle, über die Python-Skripts Dokumentteile über die Python APIs von RaptorXML aufrufen und verarbeiten können. Skripts können über CLI oder HTTP gesendet werden

Im Diagramm unten sehen Sie, wie RaptorXML über seine Schnittstellen aufgerufen wird.



Beachten Sie, dass die Verbindung zu den Server-Editionen an der COM-, Java- und .NET-Schnittstelle mit Hilfe des HTTP-Protokolls erfolgt. Python-Skripts können über die Befehlszeilenschnittstelle und die HTTP-Schnittstelle an die Server-Editionen gesendet werden.

Befehlszeilenschnittstelle (CLI)
Ermöglicht die Validierung von XML-Dateien (und anderen Dokumenten), die XSLT-Transformation und XQuery-Ausführung über die Befehlszeilenschnittstelle. Nähere Informationen dazu finden Sie

im Abschnitt Befehlszeile.

HTTP-Schnittstelle
Alle Funktionalitäten der Server-Editionen können über eine HTTP-Schnittstelle aufgerufen werden. Client-Requests werden im JSON-Format gesendet. Jedem Request wird ein Auftragsverzeichnis auf dem Server zugewiesen, in dem die Ausgabedateien gespeichert werden. Die Server-Antworten an den Client enthalten alle relevanten Informationen zum Auftrag. Siehe dazu Abschnitt HTTP-Schnittstelle.

Python-Schnittstelle
Zusammen mit einem CLI-Befehl oder einem HTTP-Request kann ein Python-Skript gesendet werden, das auf das/die im Befehl oder Request angegebene(n) Dokument(e) zugreift. Der Zugriff auf das Dokument erfolgt über Python APIs für XML, XSD und XBRL. Eine Beschreibung zur Verwendung und zu den APIs finden Sie im Abschnitt Python-Schnittstelle.

COM-Schnittstelle
RaptorXML kann über die COM-Schnittstelle verwendet werden und kann daher von Applikationen und Skripting-Sprachen, die COM unterstützen, verwendet werden. Unterstützung für die COM-Schnittstelle wurde für die Raw- und Dispatch-Schnittstelle implementiert. Die Input-Daten können als Dateien oder Textstrings in Skripts und in Applikationsdaten bereitgestellt werden.

Java-Schnittstelle
RaptorXML-Funktionalitäten stehen als in Java-Programmen verwendbare Java-Klassen zur Verfügung. So gibt es z.B. Java-Klassen zur XML-Validierung, XSLT-Transformation und XQuery-Ausführung.

.NET-Schnittstelle
Als Wrapper rund um RaptorXML wurde eine DLL-Datei verwendet. Mit Hilfe dieser Datei können .NET-Benutzer RaptorXML-Funktionen verwenden. RaptorXML bietet eine von Altova signierte primäre Interop-Assembly. Die Input-Daten können als Dateien oder als Textstrings in Skripts und Applikationsdatei bereitgestellt werden.

© 2014 Altova GmbH

1.2 Systemanforderungen

RaptorXML Server läuft auf den folgenden Betriebssystemen:

- *Windows*
  Windows XP (SP2 für x64; SP3 für x86), Windows Vista, Windows 7, Windows 8 oder neuer

- *Windows Server*
  Windows Server 2008 R2 oder neuer

- ***Linux***
  - CentOS 6 oder neuer
  - RedHat 6 oder neuer
  - Debian 6 oder neuer
  - Ubuntu 12.04 oder neuer

  Beachten Sie, dass die unter GNU GPL und LGPL verfügbare Qt Library (Version 4 oder höher) installiert sein muss.

- ***Mac OS X***
  Mac OS X 10.7 oder neuer

RaptorXML steht sowohl für 32-Bit- als auch für 64-Bit-Rechner zur Verfügung. Dabei handelt es sich um x86- und amd64- (x86-64) Instruction-Set-basierte Kerne: Intel Core i5, i7, XEON E5. Um RaptorXML über eine COM-Schnittstelle verwenden zu können, muss der Benutzer Rechte zur Benutzung der COM-Schnittstelle haben, damit er die Applikation registrieren kann und die entsprechenden Applikationen und/oder Skripts ausführen kann.

1.3 Funktionalitäten

RaptorXML enthält die unten aufgelisteten Funktionalitäten. Die meisten Funktionalitäten können sowohl über die Befehlszeilenschnittstelle als auch die COM-Schnittstelle verwendet werden. Ein bedeutender Unterschied ist, dass bei Verwendung der COM-Schnittstelle unter Windows Dokumente über die Applikation oder Skripting-Code anhand von Textstrings konstruiert werden können (anstatt XML-, DTD-, XML-Schema-, XSLT- oder XQuery-Dateien zu referenzieren).

XML-Validierung

- Validierung des bereitgestellten XML-Dokuments anhand einer internen oder externen DTD oder eines internen oder externen Schemas.
- Überprüfung der Wohlgeformtheit von XML-, DTD-, XML-Schema, XSLT- und XQuery-Dokumenten.

XSLT-Transformationen

- Transformierung von XML-Dateien anhand des bereitgestellten XSLT 1.0-, 2.0- oder 3.0-Dokuments.
- XML- und XSLT-Dokumente können als Datei (über eine URL) oder bei Verwendung der COM-Schnittstelle als Textstring bereitgestellt werden.
- Rückgabe der Ausgabe als Datei (in einem definierten Ordner) oder bei Verwendung der COM-Schnittstelle als Textstring.
- XSLT-Parameter können über die Befehlszeile und über die COM-Schnittstelle geliefert werden.
- Spezialisierte Verarbeitung dank Altova- sowie XBRL-, Java- und .NET-Erweiterungsfunktionen. Dies ermöglicht z.B. die Erstellung von Diagrammen und Barcodes in Ausgabedokumenten.

XQuery-Ausführung

- Ausführung von XQuery 1.0- und 3.0-Dokumenten
- XQuery- und XML-Dokumente können als Datei (über eine URL) oder bei Verwendung der COM-Schnittstelle als Textstring bereitgestellt werden.
- Rückgabe der Ausgabe als Datei (in einem definierten Ordner) oder bei Verwendung der COM-Schnittstelle als Textstring.
- Bereitstellung externer XQuery-Variablen über die Befehlszeile und die COM-Schnittstelle.
- Inkludiert in den Serialisierungsoptionen sind: Ausgabekodierung, Ausgabemethode (ob in XML, XHTML, HTML oder Text), Weglassen der XML-Deklaration und Einrückung.

Hochleistungsfunktionen

- Extrem hohe Verarbeitungsgeschwindigkeit dank optimiertem Code
  - Native Instruction-Set Implementierungen
  - 32-Bit- und 64-Bit-Version
- Extrem niedriger Arbeitsspeicherbedarf
  - extrem kompakte speicherresidente Darstellung des XML Information Set
  - Streaming der Instanzvalidierung
- Plattformübergreifende Funktionalitäten
- Hochgradig skalierbarer Code für Multi-CPU/Multi-Core/Parallel Computing

 © 2014 Altova GmbH

- Paralleles Laden, Validierung und Verarbeiten entsprechend dem Design

Funktionen für Entwickler

- Ausgezeichnete Fehlerberichte
- Windows Server-Modus und Unix Daemon-Modus (über Befehlszeilenoptionen)
- Inkludierter Python 3.x Interpreter für das Skripting
- COM API auf Windows-Betriebssystemen
- Java API auf allen Systemen
- XPath-Erweiterungsfunktionen Java, .NET, XBRL, & mehr
- Streaming-Serialisierung
- Integrierter HTTP-Server mit REST-Validierungs-API

Nähere Informationen dazu finden Sie im Abschnitt Unterstützte Spezifikationen und auf der Altova Website.

1.4 Unterstützte Spezifikationen

RaptorXML unterstützt die folgenden Spezifikationen.

W3C Recommendations
Website: World Wide Web Consortium (W3C)

- Extensible Markup Language (XML) 1.0 (Fifth Edition)
- Extensible Markup Language (XML) 1.1 (Second Edition)
- Namespaces in XML 1.0 (Third Edition)
- Namespaces in XML 1.1 (Second Edition)
- XML Information Set (Second Edition)
- XML Base (Second Edition)
- XML Inclusions (XInclude) Version 1.0 (Second Edition)
- XML Linking Language (XLink) Version 1.0
- XML Schema Part 1: Structures Second Edition
- XML Schema Part 2: Datatypes Second Edition
- W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures
- W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes
- XPointer Framework
- XPointer xmlns() Scheme
- XPointer element() Scheme
- XML Path Language (XPath) Version 1.0
- XSL Transformations (XSLT) Version 1.0
- XML Path Language (XPath) 2.0 (Second Edition)
- XSL Transformations (XSLT) Version 2.0
- XQuery 1.0: An XML Query Language (Second Edition)
- XQuery 1.0 and XPath 2.0 Functions and Operators (Second Edition)
- XML Path Language (XPath) 3.0

W3C Working Drafts & Candidate Recommendations
Website: World Wide Web Consortium (W3C)

- XSL Transformations (XSLT) Version 3.0
- XQuery 3.0: An XML Query Language
- XPath and XQuery Functions and Operators 3.0

OASIS Standards
Website: OASIS Standards

- XML Catalogs V 1.1 - OASIS Standard V1.1

© 2014 Altova GmbH

Kapitel 2

Einrichten von RaptorXML

2 Einrichten von RaptorXML

Dieser Abschnitt enthält Anleitungen zum Einrichten von RaptorXML Server. Es sind darin die folgenden Schritte beschrieben:

- Installation und Lizenzierung von RaptorXML auf Windows-, auf Linux- und Mac OS X-Systemen.
- Verwendung von XML-Katalogen.
- Arbeiten mit globalen Altova-Ressourcen.
- Sicherheitsfragen im Zusammenhang mit RaptorXML.

RaptorXML hat spezielle Optionen, die XML-Kataloge und globale Altova-Ressourcen unterstützen. Beide verbessern die Portabilität und Modularität. Sie können diese Funktionalitäten daher in Ihrer eigenen Umgebung vorteilhaft einsetzen.

Anmerkung: Sicherheitsfragen und das Einrichten wichtiger Sicherheitsmaßnahmen werden im Abschnitt Sicherheitsfragen beschrieben.

© 2014 Altova GmbH

2.1 Einrichten unter Windows

In diesem Abschnitt werden die Installation und Lizenzierung von RaptorXML Server auf Windows-Systemen beschrieben.

Installation unter Windows

- Systemanforderungen
- Installation von RaptorXML Server
- Altova LicenseServer
- LicenseServer-Versionen
- Testlizenz
- Pfad des Applikationsordners

Lizenzierung unter Windows

- Starten von ServiceController
- Starten von LicenseServer
- Starten von RaptorXML Server
- Registrieren von RaptorXML Server
- Zuweisen einer Lizenz

2.1.1 Installation unter Windows

Altova RaptorXML Server ist für die Installation auf Windows-Systemen verfügbar. Im Folgenden finden Sie eine Anleitung zur Installation und Konfiguration.

- Systemanforderungen
  - *Windows*
    Windows XP (SP2 für x64; SP3 für x86), Windows Vista, Windows 7, Windows 8 oder neuer

  - *Windows Server*
    Windows Server 2008 R2 oder neuer

- Installation von RaptorXML Server
  RaptorXML Server kann folgendermaßen auf Windows-Systemen installiert werden:

  - als separates, eigenständiges Server-Produkt namens RaptorXML Server. Um RaptorXML Server zu installieren, laden Sie den RaptorXML Server Installer herunter und installieren Sie das Programm. Befolgen Sie die Anweisungen auf dem Bildschirm.
  - Als Teil des Altova FlowForce Server Installationspakets[[FlowForce Server ist eine Applikation zur Planung und zeitlichen Festsetzung verschiedener Arten von Aufträgen für Altova-Server-Produkte. Daher bietet das FlowForce Server-Paket Optionen, Altove Server-Produkte (wie <%APPNAME%>)zusammen mit FlowForce Server zu installieren. Jedes Server-Produkt wird in einem separaten Programmordner installiert.]]. Um RaptorXML Server als Teil des FlowForce Server-Pakets zu installieren, laden Sie den FlowForce Server Installer herunter und installieren Sie das Programm. Befolgen Sie die Anweisungen auf dem Bildschirm und stellen Sie sicher, dass Sie die Option zur Installation von RaptorXML Server aktiviert haben.

  Beide Produkte, FlowForce Server und RaptorXML Server, können von der Altova Website heruntergeladen werden. Die Installationsprogramme installieren die Produkte mit den erforderlichen Registrierungen. Nach der Installation befindet sich die ausführbare %APPNAME %>-Datei im folgenden Ordner:

  ```
  <ProgramFilesFolder>\Altova\RaptorXMLServer2015\bin\RaptorXML.exe
  ```

  Das Installationsprogramm führt automatisch alle Registrierungen durch, damit RaptorXML Server über eine COM-Schnittstelle, als Java-Schnittstelle und in der .NET-Umgebung verwendet werden kann. Dabei wird die ausführbare RaptorXML Server-Datei als COM-Serverobjekt registriert, die Datei `RaptorXMLLib.dll` (für die Verwendung über die Java-Schnittstelle) wird im Verzeichnis `WINDIR\system32\` installiert und die Datei `Altova.RaptorXML.dll` wird zur .NET-Referenzbibliothek hinzugefügt.

- Altova LicenseServer
  - Damit RaptorXML Server ausgeführt werden kann, muss das Programm auf einem Altova LicenseServer auf Ihrem Netzwerk registriert sein.
  - Wenn Sie RaptorXML Server oder FlowForce Server [[Nähere Informationen zu FlowForce Server finden Sie unter dem vorherigen Punkt, 'Installation von <%APPNAME%>']] auf Windows-Systemen installieren, steht eine Option zur Verfügung, mit der Altova

 © 2014 Altova GmbH

LicenseServer zusammen mit RaptorXML Server oder FlowForce Server [[Nähere Informationen zu FlowForce Server finden Sie unter dem vorherigen Punkt, 'Installation von <%APPNAME%>']] heruntergeladen und installiert werden kann.
- Wenn in Ihrem Netzwerk bereits ein Altova LicenseServer installiert ist, müssen Sie keinen weiteren installieren, es sei denn, es wird eine neuere Version von Altova LicenseServer benötigt (*Siehe nächster Punkt LicenseServer-Versionen*.)
- Aktivieren oder deaktivieren Sie bei der Installation von RaptorXML Server oder FlowForce Server [[Nähere Informationen zu FlowForce Server finden Sie unter dem vorherigen Punkt, 'Installation von <%APPNAME%>']] je nach Bedarf die Option zur Installation von Altova LicenseServer.

Nähere Informationen zum Registrieren und Lizenzieren von RaptorXML Server auf dem Altova LicenseServer und zur Lizenzierung finden Sie im Abschnitt Lizenzierung unter Windows.

▼ LicenseServer-Versionen

- Altova Server-Produkte müssen entweder mit der zum Zeitpunkt der Release von RaptorXML Server aktuellen Version von LicenseServer oder einer späteren Version von LicenseServer lizenziert werden.
- Welche Version von LicenseServer für eine bestimmte Version von RaptorXML Server geeignet ist, wird während der Installation von RaptorXML Server angezeigt. Sie können diese Version von LicenseServer zusammen mit RaptorXML Server installieren oder Sie können LicenseServer separat installieren.
- Bevor Sie eine neuere Version von LicenseServer installieren, muss eine eventuell vorhandene ältere Version deinstalliert werden. Das Installationsprogramm von LicenseServer führt dies bei Vorhandensein einer älteren Version automatisch durch.
- LicenseServer-Versionen sind rückwärts kompatibel. Sie funktionieren auch mit älteren Versionen von RaptorXML Server.
- Wenn Sie eine neue Version von RaptorXML Server installieren und Ihre installierte Version von LicenseServer älter als der dafür vorgesehene LicenseServer ist, installieren Sie die neueste auf der Altova Website verfügbare Version.
- Bei der Deinstallation werden alle Registrierungs- und Lizenzierungsinformationen aus der älteren LicenseServer-Version in einer Datenbank auf Ihrem Server gespeichert. Diese Daten werden automatisch in die neuere Version importiert.
- Sie finden die Versionsnummer des aktuell installierten LicenseServer am unteren Rand der LicenseServer Konfigurationsseite (alle Register).

Aktuelle Version: 1.11

▼ Testlizenz

Bei der Installation haben Sie die Option, eine 30-Tage-Testlizenz für RaptorXML Server anzufordern. Nach Absendung der Anforderung erhalten Sie an die von Ihnen registrierte E-Mail-Adresse eine Testlizenz zugesandt.

▼ Pfad des Applikationsordners

Die Applikation wird im folgenden Ordner installiert:

Windows XP	`C:\Programme\Altova\`
Windows Vista, Windows 7/8	`C:\Programme\Altova\`

32-Bit-Version auf 64-Bit OS	`C:\Programme (x86)\Altova\`

© 2014 Altova GmbH

2.1.2 Lizenzierung unter Windows

Damit RaptorXML Server ausgeführt werden kann, muss das Programm auf einem LicenseServer lizenziert werden. Eine Anleitung zur Lizenzierung von RaptorXML Server finden Sie weiter unten.

1. **Registrieren von RaptorXML Server** auf LicenseServer. Die Registrierung erfolgt über RaptorXML Server.
2. **Zuweisen einer Lizenz** zu RaptorXML Server. Die Zuweisung der Lizenz erfolgt über LicenseServer.

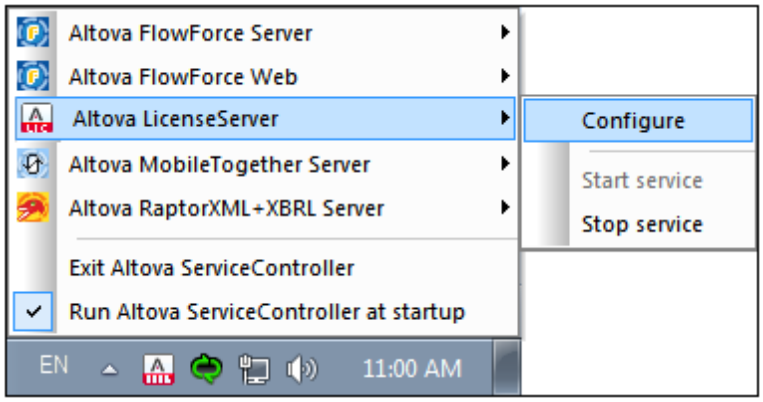
Im Folgenden sind die einzelnen Schritte beschrieben.

▼ Starten von ServiceController
Der ServiceController wird gestartet, damit Sie damit LicenseServer und RaptorXML Server starten können.

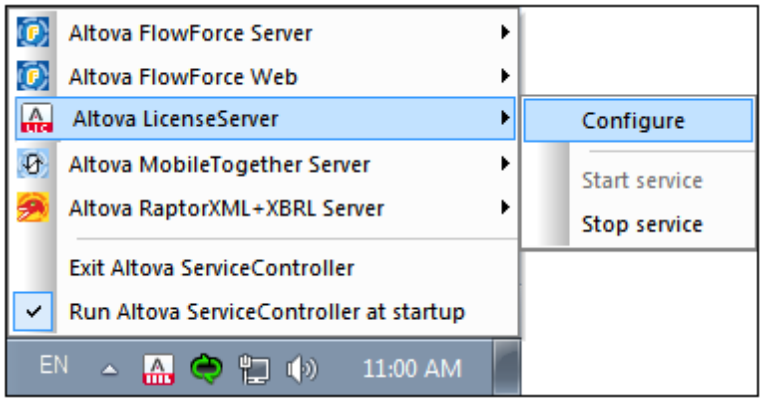
Der Altova ServiceController (in der Folge ServiceController genannt) ist eine Applikation, mit der Sie Altova-Dienste **auf Windows-Systemen** starten, beenden und konfigurieren können.

ServiceController wird mit Altova LicenseServer und jedem *Altova Server-Produkt, das ServiceController verwendet* FlowForce Server, RaptorXML(+XBRL) Server und Mobile Together Server, installiert und kann durch Klicken auf **Start | Altova LicenseServer | Altova ServiceController** gestartet werden. (Dieser Befehl steht auch in den **Startmenüordnern** der *Altova Server-Produkten, die als Dienst installiert wurden* (FlowForce Server, RaptorXML(+XBRL) Server und Mobile Together Server), zur Verfügung). Nachdem der ServiceController gestartet wurde, kann er über die Task-Leiste aufgerufen werden (*Abbildung unten*).

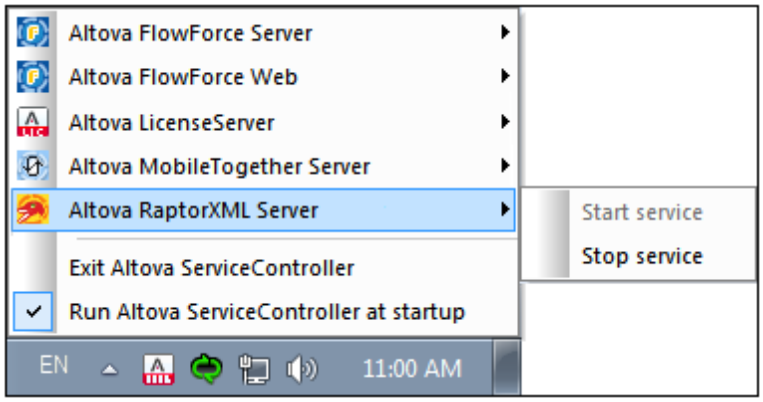
Damit der ServiceController automatisch ausgeführt wird, nachdem sich der Benutzer im System angemeldet hat, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol, um das **ServiceController-**Menü (*Abbildung unten*) aufzurufen und aktivieren Sie anschließend den Befehl **Run Altova ServiceController at Startup**. (Dieser Befehl ist standardmäßig aktiv). Um den ServiceController zu beenden, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol und klicken Sie im Menü, das daraufhin angezeigt wird (*Abbildung unten*) auf **Exit Altova ServiceController**.



- Starten von LicenseServer
  Um LicenseServer zu starten, klicken Sie in der Task-Leiste auf das Altova **ServiceController**-Symbol und platzieren Sie den Mauszeiger im angezeigten Menü über **Altova LicenseServer** (*Abbildung unten*) und wählen Sie im Untermenü von Altova LicenseServer den Befehl **Start Service**. Wenn LicenseServer bereits ausgeführt wird, ist die Option *Start Service* deaktiviert.



- Starten von RaptorXML Server
  Um RaptorXML Server zu starten, klicken Sie in der Task-Leiste auf das Altova **ServiceController**-Symbol und platzieren Sie den Mauszeiger im angezeigten Menü über **Altova RaptorXML Server** (*Abbildung unten*) und wählen Sie im Untermenü von Altova RaptorXML Server den Befehl **Start Service**. Wenn RaptorXML Server bereits ausgeführt wird, ist die Option *Start Service* deaktiviert.



- Registrieren von RaptorXML Server

  - *Registrieren von RaptorXML Server über FlowForce Server*
    Wenn RaptorXML Server im Rahmen der FlowForce Server Installation installiert wurde, wird auch RaptorXML Server automatisch auf LicenseServer registriert. Eine Beschreibung zum Registrieren von FlowForce Server finden Sie in der Dokumentation zu Altova FlowForce Server. Dabei werden im Grunde die folgenden Schritte durchgeführt: (i) Starten von Altova FlowForce Web als Dienst über ServiceController (*siehe vorheriger Punkt*); (ii) Eingabe des Passworts zum Aufrufen der Setup-Seite; (iii) Auswahl des LicenseServer-Namens bzw. -Adresse und klicken auf **Auf LicenseServer registrieren**.

    Nachdem Sie das Produkt registriert haben, können Sie RaptorXML Server über die Konfigurationsseite von LicenseServer über das Register Server Management eine Lizenz zuweisen.

  - *Registrieren einer eigenständigen Version von RaptorXML Server*
    Registrieren Sie RaptorXML Server über:

    - das CLI. Verwenden Sie dazu den Befehl `licenseserver`:
      ```
      RaptorXML licenseserver [options] ServerName-Oder-IP-Adresse
      ```
      Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist:
      ```
      RaptorXML licenseserver localhost
      ```

    Nachdem Sie RaptorXML Server erfolgreich registriert haben, können Sie dem Produkt nun über das Register Server Management der LicenseServer-Konfigurationsseite eine Lizenz zuweisen.

- Zuweisen einer Lizenz
  Nachdem Sie RaptorXML Server erfolgreich registriert haben, wird die Applikation auf der Konfigurationsseite von LicenseServer auf dem Register Server Management aufgelistet. Sie können nun zu LicenseServer wechseln und RaptorXML Server eine Lizenz zuweisen.

  Anmerkung zu Prozessorkernen und Lizenzen
  Die Lizenzierung von Altova Server-Produkten basiert *mit Ausnahme von MobileTogether Server*** auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor hat vier Kerne, ein

Hexa-Core-Prozessor hat sechs Kerne, usw. Die Anzahl der für ein Produkt auf einem bestimmten Server-Rechner lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Server verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt.

Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie eine Lizenz für mindestens acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen Octa-Core-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit eine großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Lizenz kann immer nur für einen Client gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität noch nicht ausgeschöpft ist. Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

*** MobileTogether Server-Lizenzen werden auf Basis der Anzahl der Benutzer, d.h. auf Basis der Clients, die auf MobileTogether Server zugreifen, zugewiesen.

 © 2014 Altova GmbH

2.2 Einrichten unter Linux

In diesem Abschnitt werden die Installation und Lizenzierung von RaptorXML Server auf Linux-Systemen (Debian, Ubuntu, CentOS, RedHat) beschrieben.

Installation unter Linux

- Systemanforderungen
- Anmerkungen zum Root-Benutzer
- Deinstallieren alter Versionen von Altova Server-Produkten
- Herunterladen des Linux-Pakets
- Installieren von RaptorXML Server
- Altova LicenseServer
- LicenseServer-Versionen
- Testlizenz

Lizenzierung unter Linux

- Anmerkung zum Root-Benutzer
- Starten von LicenseServer
- Starten von RaptorXML Server
- Registrieren von RaptorXML Server
- Zuweisen einer Lizenz

2.2.1 Installation unter Linux

RaptorXML Server steht für die Installation auf Linux-Systemen zur Verfügung. Im Folgenden finden Sie eine Beschreibung der Installation und Konfiguration des Produkts.

- Systemanforderungen
  - ***Linux***
    - CentOS 6 oder neuer
    - RedHat 6 oder neuer
    - Debian 6 oder neuer
    - Ubuntu 12.04 oder neuer

    Beachten Sie, dass die unter GNU GPL und LGPL verfügbare Qt Library (Version 4 oder höher) installiert sein muss.

- FlowForce Server-Integration
  Wenn Sie RaptorXML Server zusammen mit FlowForce Server installieren, sollten Sie zuerst FlowForce Server installieren. Führen Sie andernfalls nach Installation von sowohl RaptorXML Server als auch FlowForce Server den folgenden Befehl aus:

```
cp /opt/Altova/RaptorXMLServer2015/etc/*.tool /opt/Altova/
FlowForceServer2015/tools
```

  Dieser Befehl kopiert die **.tool**-Datei aus dem Verzeichnis **/etc** von RaptorXML Server in das FlowForce Server **/tools**-Verzeichnis. Die Datei **.tool** wird von FlowForce Server benötigt; sie enthält den Pfad zur ausführbaren RaptorXML Server Datei. Sie müssen diesen Befehl nicht ausführen, wenn Sie FlowForce Server vor RaptorXML Server installieren.

- Anmerkung zum Root-Benutzer
  Um RaptorXML Server installieren zu können, benötigen Sie Administratorrechte (Root-Rechte). Wenn Sie als `root`, angemeldet sind, können Sie in den unten aufgelisteten Befehlen das Schlüsselwort `sudo` weglassen.

- Deinstallieren einer alten Version von Altova Server-Produkten
  Wenn Sie eine Vorgängerversion deinstallieren müssen, gehen Sie folgendermaßen vor. Überprüfen Sie in der Linux-Befehlszeile (Command Line Interface = CLI), welche Altova Server-Produkte installiert sind. Verwenden Sie dazu den Befehl:

  `[Debian, Ubuntu]:` **`dpkg --list | grep Altova`**
  `[CentOS, RedHat]:` **`rpm -qa | grep server`**

  Falls RaptorXML Server nicht installiert ist, beginnen Sie mit der Installation, wie dies weiter unten unter *Installieren von RaptorXML Server* dokumentiert ist.

  Falls RaptorXML Server installiert ist und Sie eine neuere Version von RaptorXML Server installieren möchten, deinstallieren Sie die alte Version mit dem Befehl:

  `[Debian, Ubuntu]:` **`sudo dpkg --remove raptorxmlserver`**
  `[CentOS, RedHat]:` **`sudo rpm -e raptorxmlserver`**

 © 2014 Altova GmbH

Falls Sie eine neuere Version von Altova LicenseServer installieren möchten, deinstallieren Sie die alte Version mit dem Befehl:

```
[Debian, Ubuntu]:   sudo dpkg --remove licenseserver
[CentOS, RedHat]:   sudo rpm -e licenseserver
```

- Herunterladen des Linux-Pakets
  Auf der Altova Website stehen RaptorXML Server-Installationspakete für die folgenden Linux-Systeme zur Verfügung.

Distribution	Paketerweiterung
Debian 6 und höher	.deb
Ubuntu12.04 und höher	.deb
CentOS 6 und höher	.rpm
RedHat 6 und höher	.rpm

  Nachdem Sie das Linux-Paket von der Altova Website heruntergeladen haben, kopieren Sie das Paket in ein beliebiges Verzeichnis auf dem Linux-System. Da Sie zum Ausführen von RaptorXML Server einen Altova LicenseServer benötigen, sollten Sie gleichzeitig mit RaptorXML Server auch LicenseServer von der Altova Website herunterladen, anstatt LicenseServer erst später herunterzuladen.

- Installieren von RaptorXML Server
  Wechseln Sie in einem Terminal-Fenster zu dem Verzeichnis, in das Sie das Linux-Paket kopiert haben. Wenn Sie es z.B. in ein Benutzerverzeichnis namens `MyAltova` (das sich z.B. im Verzeichnis `/home/User` befindet) kopiert haben, dann wechseln Sie folgendermaßen zu diesem Verzeichnis:

```
cd /home/User/MyAltova
```

  Installieren Sie RaptorXML Server mit dem folgenden Befehl:

```
[Debian]:   sudo dpkg --install raptorxmlserver-2015-debian.deb
[Ubuntu]:   sudo dpkg --install raptorxmlserver-2015-ubuntu.deb
[CentOS]:   sudo rpm -ivh raptorxmlserver-2015-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh raptorxmlserver-2015-1.x86_64.rpm
```

  Das RaptorXML Server-Paket wird im folgenden Ordner installiert:

```
/opt/Altova/RaptorXMLServer2015
```

- Altova LicenseServer
  Damit ein Altova Server-Produkt - darunter auch RaptorXML Server - ausgeführt werden kann, muss es auf einem Altova LicenseServer auf Ihrem Netzwerk registriert sein.

  Auf Linux-Systemen muss Altova LicenseServer separat installiert werden. Laden Sie Altova LicenseServer von der Altova Website herunter und kopieren Sie das Paket in ein beliebiges Verzeichnis auf dem Linux-System. Installieren Sie es genau wie RaptorXML Server (*siehe vorheriger Schritt*).

```
[Debian]:   sudo dpkg --install licenseserver-1.11-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-1.11-ubuntu.deb
[CentOS]:   sudo rpm -ivh licenseserver-1.11-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh licenseserver-1.11-1.x86_64.rpm
```

Das LicenseServer-Paket wird im folgenden Ordner installiert:

```
/opt/Altova/LicenseServer
```

Eine Anleitung, wie Sie RaptorXML Server auf dem Altova LicenseServer registrieren und lizenzieren, finden Sie im Abschnitt Lizenzierung unter Linux.

▼ LicenseServer-Versionen

- Altova Server-Produkte müssen entweder mit der zum Zeitpunkt der Release von RaptorXML Server aktuellen Version von LicenseServer oder einer späteren Version von LicenseServer lizenziert werden.
- Welche Version von LicenseServer für eine bestimmte Version von RaptorXML Server geeignet ist, wird während der Installation von RaptorXML Server angezeigt. Sie können diese Version von LicenseServer zusammen mit RaptorXML Server installieren oder Sie können LicenseServer separat installieren.
- Bevor Sie eine neuere Version von LicenseServer installieren, muss eine eventuell vorhandene ältere Version deinstalliert werden. Das Installationsprogramm von LicenseServer führt dies bei Vorhandensein einer älteren Version automatisch durch.
- LicenseServer-Versionen sind rückwärts kompatibel. Sie funktionieren auch mit älteren Versionen von RaptorXML Server.
- Wenn Sie eine neue Version von RaptorXML Server installieren und Ihre installierte Version von LicenseServer älter als der dafür vorgesehene LicenseServer ist, installieren Sie die neueste auf der Altova Website verfügbare Version.
- Bei der Deinstallation werden alle Registrierungs- und Lizenzierungsinformationen aus der älteren LicenseServer-Version in einer Datenbank auf Ihrem Server gespeichert. Diese Daten werden automatisch in die neuere Version importiert.
- Sie finden die Versionsnummer des aktuell installierten LicenseServer am unteren Rand der LicenseServer Konfigurationsseite (alle Register).

Aktuelle Version: 1.11

▼ Testlizenz

Bei der Installation haben Sie die Option, eine 30-Tage-Testlizenz für RaptorXML Server anzufordern. Nach Absendung der Anforderung erhalten Sie an die von Ihnen registrierte E-Mail-Adresse eine Testlizenz zugesandt.

 © 2014 Altova GmbH

2.2.2 Lizenzierung unter Linux

Damit RaptorXML Server ausgeführt werden kann, muss das Programm auf einem Altova LicenseServer lizenziert werden. Die Lizenzierung erfolgt in zwei Schritten:

1. **Registrieren von RaptorXML Server** auf LicenseServer. Die Registrierung erfolgt über RaptorXML Server.
2. **Zuweisen einer Lizenz** zu RaptorXML Server. Die Zuweisung der Lizenz erfolgt über LicenseServer.

Im Folgenden sind die einzelnen Schritte beschrieben.

▼ Anmerkung zum Root-Benutzer

Damit Sie RaptorXML Server auf dem LicenseServer installieren können, benötigen Sie Administratorrechte (Root-Rechte). Die Installation muss daher als Root-Benutzer durchgeführt werden. Wenn Sie als `root` angemeldet sind, können Sie das Schlüsselwort `sudo` aus der Liste der unten stehenden Befehle weg lassen.

▼ Starten von LicenseServer

Damit RaptorXML Server auf dem LicenseServer korrekt registriert und lizenziert werden kann, muss LicenseServer als Daemon im Netzwerk ausgeführt werden. Starten Sie LicenseServer mit dem folgenden Befehl als Daemon:

```
[Debian]:   sudo /etc/init.d/licenseserver start
[Ubuntu]:   sudo initctl start licenseserver
[CentOS]:   sudo initctl start licenseserver
[RedHat]:   sudo initctl start licenseserver
```

Um LicenseServer zu beenden, ersetzen Sie **`start`** im obigen Befehl durch **`stop`**. Beispiel:

```
sudo /etc/init.d/licenseserver stop
```

▼ Starten von RaptorXML Server

Starten Sie RaptorXML Server mit dem folgenden Befehl als Daemon:

```
[Debian]:   sudo /etc/init.d/raptorxml start
[Ubuntu]:   sudo initctl start raptorxml
[CentOS]:   sudo initctl start raptorxml
[RedHat]:   sudo initctl start raptorxml
```

▼ Registrieren von RaptorXML Server

Registrieren Sie RaptorXML Server über:

- das CLI. Verwenden Sie dazu den Befehl `licenseserver`:

  ```
  sudo /opt/Altova/RaptorXMLServer2015/bin/raptorxml licenseserver
  [options] ServerName-Oder-IP-Adresse
  ```

  Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist:

  ```
  sudo /opt/Altova/RaptorXMLServer2015/bin/raptorxml licenseserver
  ```

```
localhost
```

Im obigen Befehl ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie außerdem, dass der Pfad der ausführbaren RaptorXML Server Datei folgendermaßen lautet:

```
/opt/Altova/RaptorXMLServer2015/bin/
```

Nachdem Sie RaptorXML Server erfolgreich registriert haben, können Sie dem Produkt nun über das Register Server Management der LicenseServer-Konfigurationsseite eine Lizenz zuweisen.

▼ Zuweisen einer Lizenz

Nachdem Sie RaptorXML Server erfolgreich registriert haben, wird das Produkt auf der Konfigurationsseite von LicenseServer auf dem Register Server Management aufgelistet. Sie können LicenseServer nun aufrufen und RaptorXML Server eine Lizenz zuweisen.

Anmerkung zu Prozessorkernen und Lizenzen

Die Lizenzierung von Altova Server-Produkten basiert *mit Ausnahme von MobileTogether Server**** auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor hat vier Kerne, ein Hexa-Core-Prozessor hat sechs Kerne, usw. Die Anzahl der für ein Produkt auf einem bestimmten Server-Rechner lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Server verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt.

Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie eine Lizenz für mindestens acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen Octa-Core-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit eine großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Lizenz kann immer nur für einen Client gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität noch nicht ausgeschöpft ist. Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

*** MobileTogether Server-Lizenzen werden auf Basis der Anzahl der Benutzer, d.h. auf Basis der Clients, die auf MobileTogether Server zugreifen, zugewiesen.

 © 2014 Altova GmbH

2.3 Einrichten auf Mac OS X

In diesem Abschnitt werden die Installation und Lizenzierung von RaptorXML Server auf Mac OS X-Systemen beschrieben.

Installation unter Mac OS X

- Systemanforderungen
- Anmerkung zu Root-Benutzern
- Deinstallieren alter Versionen von Altova Server-Produkten
- Herunterladen des Mac OS X-Pakets
- Installieren von RaptorXML Server
- Altova LicenseServer
- LicenseServer-Versionen
- Testlizenz

Licenzierung unter Mac OS X

- Anmerkung zu Root-Benutzern
- Starten von LicenseServer
- Starten von RaptorXML Server
- Registrieren von RaptorXML Server
- Zuweisen einer Lizenz

2.3.1 Installation auf Mac OS X

RaptorXML Server steht für die Installation auf Mac OS X-Systemen zur Verfügung. Im Folgenden finden Sie eine Beschreibung der Installation und Konfiguration.

- Systemanforderungen
  - ***Mac OS X***
    Mac OS X 10.7 oder neuer

- FlowForce Server-Integration
  Wenn Sie RaptorXML Server zusammen mit FlowForce Server installieren, sollten Sie zuerst FlowForce Server installieren. Führen Sie andernfalls nach Installation von sowohl RaptorXML Server als auch FlowForce Server den folgenden Befehl aus:

```
cp /usr/local/Altova/RaptorXMLServer2015/etc/*.tool /usr/local/Altova/
FlowForceServer2015/tools
```

  Dieser Befehl kopiert die **.tool**-Datei aus dem Verzeichnis **/etc** von RaptorXML Server in das FlowForce Server **/tools**-Verzeichnis. Die Datei **.tool** wird von FlowForce Server benötigt; sie enthält den Pfad zur ausführbaren RaptorXML Server Datei. Sie müssen diesen Befehl nicht ausführen, wenn Sie FlowForce Server vor RaptorXML Server installieren.

- Anmerkung zu Root-Benutzern
  Um RaptorXML Server installieren zu können, benötigen Sie Administratorrechte (Root-Rechte). Wenn Sie als `root`, angemeldet sind, können Sie in den unten aufgelisteten Befehlen das Schlüsselwort `sudo` weglassen.

- Deinstallieren alter Versionen von Altova Server-Produken
  Bevor Sie RaptorXML Server deinstallieren, stoppen Sie den Dienst mit dem folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.RaptorXMLServer2015.plist
```

  Um zu überprüfen, ob der Dienst gestoppt wurde, öffnen Sie den Activity Monitor Terminal und vergewissern Sie sich, dass sich RaptorXML Server nicht in der Liste befindetKlicken Sie im Applikations-Terminal mit der rechten Maustaste auf das RaptorXML Server-Symbol und wählen Sie den Befehl **Move to Trash**. Die Applikation wird daraufhin in den Papierkorb verschoben. Sie müssen die Applikation jetzt noch aus dem Ordner `usr` entfernen. Verwenden Sie dazu den folgenden Befehl:

```
sudo rm -rf /usr/local/Altova/RaptorXMLServer2015/
```

  Wenn Sie eine ältere Version von Altova LicenseServer deinstallieren müssen, stoppen Sie den Dienst zuerst mit dem folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

  Um zu überprüfen, ob der Dienst gestoppt wurde, öffnen Sie den Activity Monitor Terminal und vergewissern Sie sich, dass sich LicenseServer nicht auf der Liste befindet. Fahren Sie anschließend mit der Deinstallation fort, wie oben für RaptorXML Server beschrieben.

 © 2014 Altova GmbH

Herunterladen des Mac OS X-Pakets

Nachdem Sie das Mac OS X-Paket von der Altova Website heruntergeladen haben, kopieren Sie es in ein Verzeichnis auf dem Mac OS X-System. Da Sie zum Ausführen von RaptorXML Server einen Altova LicenseServer benötigen, sollten Sie, wenn Sie RaptorXML Server herunterladen, gleichzeitig auch den LicenseServer von der Altova Website herunterladen. Die Mac OS X Installationsdatei hat die Erweiterung `.pkg`.

Installieren von RaptorXML Server

Wechseln Sie in einem Terminal-Fenster zu dem Verzeichnis, in das Sie die Installationsdatei kopiert haben und doppelklicken Sie auf die Datei. Befolgen Sie die Anweisungen des Installationsassistenten, in dem Sie auch die Lizenzvereinbarung akzeptieren müssen, bevor Sie mit der Installation fortfahren können.

Das RaptorXML Server-Paket wird im folgenden Ordner installiert:

```
/usr/local/Altova/RaptorXMLServer2015
```

Wenn Sie im Applikations-Terminal auf das RaptorXML Server-Symbol klicken, wird die Bildschirmhilfe (diese Dokumentation) angezeigt.

Altova LicenseServer

Damit ein Altova Server-Produkt - darunter auch RaptorXML Server - ausgeführt werden kann, muss es auf einem Altova LicenseServer auf Ihrem Netzwerk lizenziert sein.

Auf Mac OS X-Systemen muss Altova LicenseServer separat installiert werden. Laden Sie das Altova LicenseServer-Paket von der Altova Website herunter und doppelklicken Sie auf das Installer-Paket, um die Installation zu starten. Befolgen Sie die Anweisungen auf dem Bildschirm. Damit die Installation fortgesetzt wird, müssen Sie die Lizenzvereinbarung akzeptieren.

Das LicenseServer-Paket wird im folgenden Ordner installiert:

```
/usr/local/Altova/LicenseServer
```

Informationen zum Registrieren von RaptorXML Server auf dem Altova LicenseServer und zum Lizenzieren des Produkts finden Sie im Abschnitt Lizenzierung unter Mac OS X.

LicenseServer-Versionen

- Altova Server-Produkte müssen entweder mit der zum Zeitpunkt der Release von RaptorXML Server aktuellen Version von LicenseServer oder einer späteren Version von LicenseServer lizenziert werden.
- Welche Version von LicenseServer für eine bestimmte Version von RaptorXML Server geeignet ist, wird während der Installation von RaptorXML Server angezeigt. Sie können diese Version von LicenseServer zusammen mit RaptorXML Server installieren oder Sie können LicenseServer separat installieren.
- Bevor Sie eine neuere Version von LicenseServer installieren, muss eine eventuell vorhandene ältere Version deinstalliert werden. Das Installationsprogramm von LicenseServer führt dies bei Vorhandensein einer älteren Version automatisch durch.
- LicenseServer-Versionen sind rückwärts kompatibel. Sie funktionieren auch mit älteren

Versionen von RaptorXML Server.

- Wenn Sie eine neue Version von RaptorXML Server installieren und Ihre installierte Version von LicenseServer älter als der dafür vorgesehene LicenseServer ist, installieren Sie die neueste auf der Altova Website verfügbare Version.
- Bei der Deinstallation werden alle Registrierungs- und Lizenzierungsinformationen aus der älteren LicenseServer-Version in einer Datenbank auf Ihrem Server gespeichert. Diese Daten werden automatisch in die neuere Version importiert.
- Sie finden die Versionsnummer des aktuell installierten LicenseServer am unteren Rand der LicenseServer Konfigurationsseite (alle Register).

Aktuelle Version: 1.11

▼ Testlizenz

Bei der Installation haben Sie die Option, eine 30-Tage-Testlizenz für RaptorXML Server anzufordern. Nach Absendung der Anforderung erhalten Sie an die von Ihnen registrierte E-Mail-Adresse eine Testlizenz zugesandt.

© 2014 Altova GmbH

2.3.2 Lizenzierung auf Mac OS X

Damit RaptorXML Server ausgeführt werden kann, muss das Programm auf einem Altova LicenseServer lizenziert werden. Die Lizenzierung erfolgt in zwei Schritten:

1. **Registrieren von RaptorXML Server** auf LicenseServer. Die Registrierung erfolgt über RaptorXML Server.
2. **Zuweisen einer Lizenz** zu RaptorXML Server. Die Zuweisung der Lizenz erfolgt über LicenseServer.

Im Folgenden sind die einzelnen Schritte beschrieben.

▼ Anmerkung zum Root-Benutzer

Damit Sie RaptorXML Server auf dem LicenseServer installieren können, benötigen Sie Administratorrechte (Root-Rechte). Die Installation muss daher als Root-Benutzer durchgeführt werden. Wenn Sie als `root` angemeldet sind, können Sie das Schlüsselwort `sudo` aus der Liste der unten stehenden Befehle weg lassen.

▼ Starten von LicenseServer

Damit RaptorXML Server auf dem LicenseServer korrekt registriert und lizenziert werden kann, muss LicenseServer als Daemon im Netzwerk ausgeführt werden. Starten Sie LicenseServer mit dem folgenden Befehl als Daemon:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Um LicenseServer zu beenden, ersetzen Sie **load** im obigen Befehl durch **unload**. Beispiel:

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.LicenseServer.plist
```

▼ Starten von RaptorXML Server

Starten Sie RaptorXML Server mit dem folgenden Befehl als Daemon:

```
sudo launchctl load /Library/LaunchDaemons/
com.altova.RaptorXMLServer2015.plist
```

Wenn Sie RaptorXML Server stoppen müssen, verwenden Sie:

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.RaptorXMLServer2015.plist
```

▼ Registrieren von RaptorXML Server

Registrieren Sie RaptorXML Server über:

- das CLI. Verwenden Sie dazu den Befehl `licenseserver`:

```
sudo /opt/Altova/RaptorXMLServer2015/bin/raptorxml licenseserver
[options] ServerName-Oder-IP-Adresse
```

Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist:

```
sudo /opt/Altova/RaptorXMLServer2015/bin/raptorxml licenseserver
localhost
```

Im obigen Befehl ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie außerdem, dass der Pfad der ausführbaren RaptorXML Server Datei folgendermaßen lautet:

```
/opt/Altova/RaptorXMLServer2015/bin/
```

Nachdem Sie RaptorXML Server erfolgreich registriert haben, können Sie dem Produkt nun über das Register Server Management der LicenseServer-Konfigurationsseite eine Lizenz zuweisen.

Zuweisen einer Lizenz

Nachdem Sie RaptorXML Server erfolgreich registriert haben, wird das Produkt auf der Konfigurationsseite von LicenseServer auf dem Register Server Management aufgelistet. Sie können LicenseServer nun aufrufen und RaptorXML Server eine Lizenz zuweisen.

Anmerkung zu Prozessorkernen und Lizenzen

Die Lizenzierung von Altova Server-Produkten basiert *mit Ausnahme von MobileTogether Server**** auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor hat vier Kerne, ein Hexa-Core-Prozessor hat sechs Kerne, usw. Die Anzahl der für ein Produkt auf einem bestimmten Server-Rechner lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Server verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt.

Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie eine Lizenz für mindestens acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen Octa-Core-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit eine großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Lizenz kann immer nur für einen Client gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität noch nicht ausgeschöpft ist. Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

*** MobileTogether Server-Lizenzen werden auf Basis der Anzahl der Benutzer, d.h. auf Basis der Clients, die auf MobileTogether Server zugreifen, zugewiesen.

 © 2014 Altova GmbH

2.4 XML-Kataloge

Mit Hilfe von XML-Katalogen können Dateien aus lokalen Ordnern aufgerufen werden, anstatt von entfernten URLs, wodurch die Verarbeitungsgeschwindigkeit erhöht und die Portabilität von Dokumenten verbessert wird, da dann nur die URIs der Katalogdateien geändert werden müssen. Nähere Informationen dazu finden Sie im Abschnitt Funktionsweise von Katalogen.

Die XML-Produkte von Altova verwenden einen Katalogmechanismus, um häufig benötigte Dateien wie DTDs und XML-Schemas schnell aufrufen und laden zu können. Dieser Katalogmechanismus kann vom Benutzer angepasst und erweitert werden und ist im Abschnitt XML-Katalogmechanismus von Altova beschrieben. Im Abschnitt Variablen für Systempfade sind Windows-Variablen für häufig benötigte Systempfade aufgelistet. Diese Variablen können in Katalogdateien verwendet werden, um häufig verwendete Ordner aufzurufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- Funktionsweise von Katalogen
- XML-Katalogmechanismus von Altova
- Variablen für Systempfade

Nähere Informationen zu Katalogen finden Sie in der XML-Katalogspezifikation.

2.4.1 Funktionsweise von Katalogen

In diesem Abschnitt werden folgende Punkte beschrieben:

- Mappen von Public und System Identifiern auf lokale URLs
- Mappen von Dateipfaden, Web URLs und Namen auf lokale URLs

Mit Hilfe von Katalogen können Aufrufe entfernter Ressourcen an eine lokale URL umgeleitet werden. Dazu werden in der Katalogdatei Public oder System Identifier, URIs oder Teile von Identifiern an die benötigte lokale URL umgeleitet.

Mappen von Public und System Identifiern auf lokale URLs
Wenn die `DOCTYPE` Deklaration einer DTD in einer XML-Datei gelesen wird, verweist der Public oder System Identifier der Deklaration auf die benötigte Ressource. Wenn der Identifier auf eine entfernte Ressource verweist oder wenn der Identifier kein Lokator ist, kann er dennoch über einen Eintrag in der Katalogdatei auf eine lokale Ressource gemappt werden.

Betrachten Sie als Beispiel die folgende SVG-Datei:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg>
  ...
</svg>
```

Ihr Public Identifier lautet: **`-//W3C//DTD SVG 1.1//EN`**
Ihr System Identifier lautet: `http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd`

Mit Hilfe eines Eintrags in einer Katalogdatei könnte der Public Identifier z.B. auf die folgende URL gemappt werden:

```
<public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
```

Oder der System Identifier könnte mit Hilfe eines Eintrags in einer Katalogdatei auf die folgende lokale URL gemappt werden:

```
<system systemId="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"
uri="schemas/svg/svg11.dtd"/>
```

Falls der Katalog einen übereinstimmenden Eintrag für den Public oder System Identifier enthält, wird die URL verwendet, auf die der Identifier gemappt ist. (Relative Pfade werden mit Verweis auf ein `xml:base` Attribut im Katalogelement, das die Umleitung enthält, aufgelöst; als Basis-URL wird die URL der Katalogdatei verwendet.) Falls der Katalog keinen übereinstimmenden Eintrag für den Public oder System Identifier enthält, wird die URL im XML-Dokument verwendet (im obigen Beispiel: `http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd`).

 © 2014 Altova GmbH

Mappen von relativen oder absoluten Dateipfaden, Web URLs oder Namen auf lokale URLs

Mit Hilfe des Elements `uri` kann ein relativer oder absoluter Dateipfad oder eine Web URL oder ein beliebiger Name auf eine lokale URL gemappt werden. Dies funktioniert folgendermaßen:

- `<uri name="doc.xslt" uri="C:\Docs\doc.xslt"/>`
- `<uri name="U:\Docs\2013\doc.xslt" uri="C:\Docs\doc.xslt"/>`
- `<uri name="http://www.altova.com/schemas/doc.xslt" uri="C:\Docs\doc.xslt"/>`
- `<uri name="foo" uri="C:\Docs\doc.xslt"/>`

Wenn der Wert `name` gefunden wird, wird er auf die im `uri` Attribut angegebene Ressource gemappt. Bei Verwendung eines anderen Katalogs kann derselbe Name auf eine andere Ressource gemappt werden. Angenommen Sie haben den folgenden Dateipfad:

```
xsi:schemaLocation="http://www.altova.com/schemas/orgchart OrgChart.xsd"
```

Normalerweise ist der URI-Teil des Attributwerts (im Beispiel oben fett gedruckt) ein Pfad zum aktuellen Schema. Wenn das Schema jedoch über einen Katalog referenziert wird, muss der URI-Teil nicht auf ein aktuelles XML-Schema verweisen, muss aber vorhanden sein, damit das Attribut `xsi:schemaLocation` lexikalisch gültig ist. So wäre z.B. der Wert `foo` für den URI-Teil des `xsi:schemaLocation` Attributwerts (anstelle von `Orgchart.xsd`) ausreichend. Das Schema wird im Katalog anhand des Namespace-Teils des `xsi:schemaLocation` Attributwerts gefunden. Im obigen Beispiel lautet der Namespace-Teil `http://www.altova.com/schemas/orgchart`.

Im Katalog würde das Schema mit dem folgenden Eintrag anhand dieses Namespace-Teils gefunden.

```
<uri name="http://www.altova.com/schemas/orgchart" uri="C:\MySchemas
\OrgChart.xsd"/>
```

Nähere Informationen zu diesen Elementen finden Sie in der XML-Katalogspezifikation.

2.4.2 Der XML-Katalogmechanismus von Altova

In diesem Abschnitt werden folgende Punkte beschrieben:

- Die root-Katalogdatei, `RootCatalog.xml`, enthält die Katalogdateien, in denen RaptorXML nachsieht.
- Altovas Katalog-Erweiterungsdateien: `CoreCatalog.xml`, `CustomCatalog.xml` und `Catalog.xml`.
- Unterstützte Kataloguntergruppe

RootCatalog.xml
Standardmäßig ruft RaptorXML die Liste der zu verwendenden Katalogdateien aus der Datei `RootCatalog.xml` (*Codefragment unten*) auf. `RootCatalog.xml` befindet sich im folgenden Ordner:

```
<ProgramFilesFolder>\Altova\RaptorXMLServer2015\etc
```

Um eine andere Datei als die Root-Katalogdatei zu verwenden, verwenden Sie in der Befehlszeile die Option `--catalog`, für die Java-Schnittstelle die Methode `setCatalog` oder die Methode `Catalog` der COM-Schnittstelle.

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
         xmlns:spy="http://www.altova.com/catalog_ext"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog
Catalog.xsd">

  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/
CustomCatalog.xml"/>
  <nextCatalog catalog="CoreCatalog.xml"/>

  <!-- Include all catalogs under common schemas folder on the first directory
level -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/Schemas"
catalog="catalog.xml" spy:depth="1"/>

  <!-- Include all catalogs under common XBRL folder on the first directory
level -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/XBRL" catalog="catalog.xml"
spy:depth="1"/>
</catalog>
```

Zusätzliche zu verwendende Katalogdateien werden jeweils in einem `nextCatalog` Element aufgelistet, von denen beliebig viele hinzugefügt werden können. Das Programm ruft jede Katalogdatei auf und löst die darin enthaltenen Mappings auf.

Beachten Sie, dass im Codefragment oben zwei Kataloge direkt referenziert sind:

CoreCatalog.xml und CustomCatalog.xml. Zusätzlich dazu werden auch Kataloge namens catalog.xml, die sich in der ersten Ebene der untergeordneten Ordner der Ordner Schemas und XBRL befinden, referenziert. (Den Wert der Variablen %AltovaCommonFolder% finden Sie im Abschnitt Variablen für Systempfade.)

In den Katalogdateien im Altova Ordner "Common" sind die vordefinierten Public und System Identifier (wie z.B. XML Schema und XHTML) auf URIs gemappt, die auf lokale Kopien der entsprechenden Schemas verweisen. Diese Schemas werden bei Installation von RaptorXML im Altova Ordner "Common" installiert.

CoreCatalog.xml, CustomCatalog.xml und Catalog.xml
Die Katalogdateien CoreCatalog.xml und CustomCatalog.xml sind im RootCatalog.xml aufgelistet und können hier nachgeschlagen werden:

- CoreCatalog.xml enthält bestimmte Altova-spezifische Mappings, um Schemas im Altova Ordner "Common" zu finden.
- CustomCatalog.xml ist eine Skelettdatei, in der Sie Ihre eigenen Mappings erstellen können. Sie können für jedes gewünschte Schema, das nicht in den Katalogdateien im Altova Ordner "Common" enthalten ist, Mappings zu CustomCatalog.xml hinzufügen. Verwenden Sie dazu die unterstützten Elemente des OASIS-Katalogmechanismus (*siehe unten*).
- Im Ordner eines bestimmten Schemas oder einer XBRL-Taxonomie gibt es im Altova Ordner "Common" eine Reihe von Catalog.xml Dateien. In jeder davon sind Public und/ oder System Identifier auf URIs gemappt, die auf lokal gespeicherte Kopien der jeweiligen Schemas verweisen.

Sowohl CoreCatalog.xml als auch CustomCatalog.xml befinden sich im Ordner<*ProgramFilesFolder*>\Altova\RaptorXMLServer2015\etc. Die catalog.xml Dateien befinden sich alle in einem eigenen Schemaordner, wobei sich diese Schemaordner innerhalb der folgenden Ordner befinden: %AltovaCommonFolder%\Schemas und %AltovaCommonFolder%\XBRL.

Unterstützte Kataloguntergruppe
Wenn Sie in einer von RaptorXML verwendeten Katalogdatei Einträge erstellen, verwenden Sie nur die folgenden Elemente der OASIS-Katalogspezifikation. Jedes der unten angeführten Elemente ist mit einer Erklärung seiner Attributwerte aufgelistet. Ausführlichere Informationen dazu finden Sie in der XML-Katalogspezifikation.

- `<public publicId="PublicID of Resource" uri="URL of local file"/>`
- `<system systemId="SystemID of Resource" uri="URL of local file"/>`
- `<uri name="filename" uri="URL of file identified by filename"/>`
- `<rewriteURI uriStartString="StartString of URI to rewrite" rewritePrefix="String to replace StartString"/>`
- `<rewriteSystem systemIdStartString="StartString of SystemID" rewritePrefix="Replacement string to locate resource locally"/>`

In Fällen, in denen es keinen Public Identifier gibt, kann der System Identifier über das Element

`system` direkt auf eine URL gemappt werden. Eine URI kann über das Element `uri` auch auf eine andere URI gemappt werden. Die Elemente `rewriteURI` und `rewriteSystem` ermöglichen das Neuschreiben des Anfangsteils einer URI bzw. eines System Identifiers. Dadurch kann der Anfang eines Dateipfads ersetzt werden, sodass ein anderes Verzeichnis als Ziel gewählt werden kann.

Anmerkung: Jedes Element kann das Attribut `xml:base` erhalten, mit dem die Basis-URI dieses Elements definiert wird. Wenn kein `xml:base`-Element vorhanden ist, wird als Basis-URI die URI der Katalogdatei verwendet.

Nähere Informationen zu diesen Elementen finden Sie in der XML-Katalogspezifikation.

© 2014 Altova GmbH

2.4.3 Variablen für Windows-Systempfade

Mit Hilfe von Shell-Umgebungsvariablen kann in Katalogdateien der Pfad zu verschiedenen Windows-Systempfaden definiert werden. Die folgenden Variablen werden unterstützt:

%AltovaCommonFolder%	C:\Program Files\Altova\Common2015
%DesktopFolder%	Vollständiger Pfad des Desktop-Ordners für den aktuellen Benutzer.
%ProgramMenuFolder%	Vollständiger Pfad zum Ordner 'Programme" für den aktuellen Benutzer.
%StartMenuFolder%	Vollständiger Pfad zum Startmenüordner für den aktuellen Benutzer.
%StartUpFolder%	Vollständiger Pfad zum Startordner für den aktuellen Benutzer.
%TemplateFolder%	Vollständiger Pfad zum Ordner "Templates" für den aktuellen Benutzer.
%AdminToolsFolder%	Vollständiger Pfad zum Dateisystemverzeichnis, in dem Verwaltungstools für den aktuellen Benutzer gespeichert sind.
%AppDataFolder%	Vollständiger Pfad zum Ordner "Anwendungsdaten" für den aktuellen Benutzer.
%CommonAppDataFolder%	Vollständiger Pfad zum Dateisystem, das die Applikationsdaten für alle Benutzer enthält.
%FavoritesFolder%	Vollständiger Pfad zum Ordners "Favoriten' für den aktuellen Benutzer.
%PersonalFolder%	Vollständiger Pfad zum persönlichen Ordner für den aktuellen Benutzer.
%SendToFolder%	Vollständiger Pfad zum Ordner SendTo für den aktuellen Benutzer.
%FontsFolder%	Vollständiger Pfad zum Ordner Systemschriftarten.
%ProgramFilesFolder%	Vollständiger Pfad zum Ordner "Programmdateien" für den aktuellen Benutzer.
%CommonFilesFolder%	Vollständiger Pfad zum Ordner "Gemeinsame Dateien" für den aktuellen Benutzer.
%WindowsFolder%	Vollständiger Pfad zum Windows-Ordner für den aktuellen Benutzer.
%SystemFolder%	Vollständiger Pfad zum Ordner "System" für den aktuellen Benutzer.

`%LocalAppDataFolder%`	Vollständiger Pfad zum Dateisystemverzeichnis, das als Datenspeicher für lokale (nicht entfernt ausgeführte) Applikationen dient.
`%MyPicturesFolder%`	Vollständiger Pfad zum Ordner "Meine Bilder".

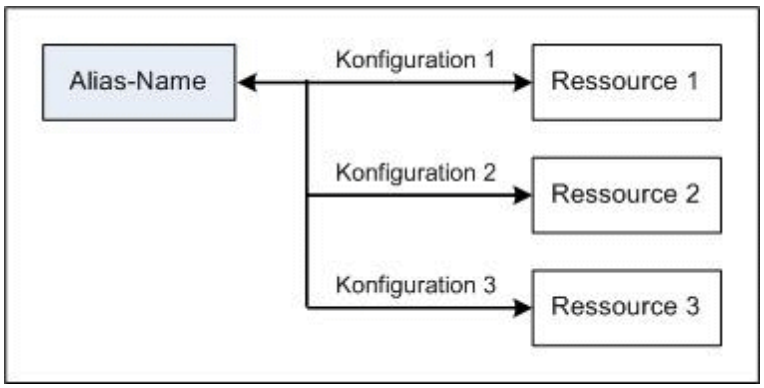
© 2014 Altova GmbH

2.5 Globale Ressourcen

In diesem Abschnitt werden folgende Punkte beschrieben:

- Informationen zu globalen Ressourcen
- Verwendung von globalen Ressourcen

Informationen zu globalen Ressourcen
In einer Datei für globale Altova-Ressourcen wird ein Alias, wie im Diagramm unten gezeigt, über verschiedene Konfigusrationen auf mehrere Ressourcen gemappt. Um eine andere Ressource aufzurufen, kann daher ein anderer Alias verwendet werden, um die Konfiguration zu wechseln.



Globale Ressourcen werden in Altova-Produkten wie z.B. Altova XMLSpy definiert und in einer XML-Datei für globale Ressourcen gespeichert. RaptorXML kann globale Ressourcen als Inputs verwenden. Dazu werden der Name und Pfad der Datei für globale Ressourcen sowie der Alias und die zu verwendende Konfiguration benötigt.

Der Vorteil der Verwendung globaler Ressourcen ist, dass die Ressource einfach durch Verwendung eines anderen Konfigurationsnamens gewechselt werden kann. Bei Verwendung von RaptorXML kann durch Angabe eines anderen Werts für die `--globalresourcesconfig | --gc` Option eine andere Ressource verwendet werden. (*Siehe Beispiel unten.*)

Verwendung globaler Ressourcen mit RaptorXML
Um eine globale Ressource als Input für einen RaptorXML-Befehl zu verwenden, sind die folgenden Parameter erforderlich:

- Die XML-Datei für globale Ressourcen (in der Befehlszeilenschnittstelle angegeben mit der Option `--globalresourcesfile | --gr`)
- Die erforderliche Konfiguration (in der Befehlszeilenschnittstelle angegeben mit der Option `--globalresourcesconfig | --gc`)
- Der Alias. Dieser kann, wo ein Dateiname erforderlich ist, direkt in der Befehlszeilenschnittstelle angegeben werden oder als Node in einer XML-Datei definiert

sein, in dem RaptorXML den Dateinamen abruft (z.B. im Attribut `xsi:schemaLocation`).

Wenn Sie z.B. `input.xml` mittels `transform.xslt` in `output.html` transformieren möchten, erfolgt dies normalerweise über die Befehlszeilenschnittstelle mit dem folgenden Befehl, in dem Dateinamen verwendet werden:

```
raptorxml xslt --input=input.xml --output=output.html transform.xslt
```

Wenn Sie jedoch eine Definition für globale Ressourcen haben, die den Alias `MyInput` über eine Konfigurationsdatei namens `FirstConfig` der Dateiressource `FirstInput.xml` zuordnet, so könnten Sie den Alias `MyInput` in der Befehlszeile folgendermaßen verwenden:

```
raptorxml xslt --input=altova://file_resource/MyInput --gr=C:
\MyGlobalResources.xml --gc=FirstConfig --output=Output.html transform.xslt
```

Angenommen, Sie haben eine andere Dateiressource namens `SecondInput.xml`, die dem Alias `MyInput` über eine Konfiguration namens `SecondConfig` zugeordnet ist, so können Sie diese Ressource verwenden, indem Sie nur die Option `--gc` des vorherigen Befehls ändern:

```
raptorxml xslt --input=altova://file_resource/MyInput --gr=C:
\MyGlobalResources.xml --gc=SecondConfig --output=Output.html transform.xslt
```

Anmerkung: Im obigen Beispiel wurde eine Dateiressource verwendet; einer Dateiressource muss das Präfix **`altova://file_resource/`** vorangestellt werden. Sie können auch globale Ressourcen verwenden, die Ordner sind. Um eine Ordnerressource zu identifizieren, verwenden Sie: **`altova://folder_resource/AliasName`**. Beachten Sie, dass Sie in der Befehlszeilenschnittstelle auch Ordnerressourcen als Teil eines Dateipfads verwenden können. Beispiel: **`altova://folder_resource/AliasName/input.xml`**.

 © 2014 Altova GmbH

2.6 Sicherheitsfragen

In diesem Abschnitt werden folgende Punkte beschrieben:

- Sicherheitsfragen im Zusammenhang mit der HTTP-Schnittstelle
- Python-Skripts sicher machen

Einige Schnittstellenfunktionalitäten von RaptorXML Server können ein Sicherheitsrisiko darstellen. Im Folgenden werden diese sowie die Lösung des Problems beschrieben.

Sicherheitsfragen im Zusammenhang mit der HTTP-Schnittstelle
Standardmäßig können Ergebnisdokumente über die HTTP-Schnittstelle in jeden durch den Client angegebenen Ordner (auf den über das HTTP-Protokoll Zugriff besteht) geschrieben werden. Beim Konfigurieren von RaptorXML Server sollte dieser Sicherheitsaspekt daher berücksichtigt werden.

Falls die Sicherheit eventuell gefährdet sein könnte oder jemand über die Schnittstelle unbefugten Zugriff erhalten könnte, kann der Server so konfiguriert werden, dass Ergebnisdokumente in ein eigenes Ausgabeverzeichnis auf dem Server selbst geschrieben werden. Dies geschieht durch Setzen der Option `server.unrestricted-filesystem-access` der Server-Konfigurationsdatei auf `false`. Wenn der Zugriff auf diese Weise eingeschränkt ist, kann der Client Ergebnisdokumente aus dem dafür vorgesehenen Ausgabeverzeichnis mit `GET` Requests herunterladen. Alternativ dazu kann ein Administrator die Ergebnisdokumentdateien vom Server in den Zielordner kopieren/laden.

Python-Skripts sicher machen
Wenn ein Python-Skript in einem Befehl über HTTP an RaptorXML Server adressiert ist, funktioniert das Skript nur, wenn es sich im vertrauenswürdigen Verzeichnis befindet. Das Skript wird vom vertrauenswürdigen Verzeichnis aus ausgeführt. Wenn Sie ein Python-Skript aus einem anderen Verzeichnis definieren, wird ein Fehler ausgegeben. Das vertrauenswürdige Verzeichnis wird in der `server.script-root-dir` Einstellung der Serverkonfigurationsdatei definiert. Wenn Sie Python-Skripts verwenden möchten, **muss** ein vertrauenswürdiges Verzeichnis definiert werden. Stellen Sie sicher, dass alle Python-Skripts, die verwendet werden sollen, in diesem Verzeichnis gespeichert werden.

Zwar werden alle vom Server für HTTP-Auftragsanforderungen generierten Ausgabedateien in das Auftragsausgabeverzeichnis (ein Unterverzeichnis von `output-root-directory`) geschrieben, doch gilt diese Einschränkung nicht für Python-Skripts, die in jeden Ordner geschrieben werden können. Der Server-Administrator muss die Python-Skripts im vertrauenswürdigen Verzeichnis auf potentielle Schwachstellen überprüfen.

Kapitel 3

Befehlszeilenschnittstelle (CLI)

3 Befehlszeilenschnittstelle (CLI)

Die ausführbare RaptorXML Datei für die Verwendung über die Befehlszeilenschnittstelle (CLI) befindet sich standardmäßig unter folgendem Pfad:

Windows `<ProgramFilesFolder>\Altova\RaptorXMLServer2015\bin\RaptorXML`
Linux `\opt\Altova\RaptorXMLServer2015\bin\raptorxml`
Mac `\usr\local\Altova\RaptorXMLServer2015\bin\raptorxml`

▼ Groß- und Kleinschreibung in der Befehlszeile

`RaptorXML` *unter Windows*
`raptorxml` *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Verwendung
Die Befehlszeilensyntax lautet:

Windows **`RaptorXML --h | --help | --version | <command> [options] [arguments]`**
Linux **`raptorxml --h | --help | --version | <command> [options] [arguments]`**
Mac **`raptorxml --h | --help | --version | <command> [options] [arguments]`**

`raptorxml`	Ruft die Applikation auf.
`raptorxml`	Ruft die Applikation auf Unix-Plattformen (Linux und Mac) auf.
`--h | --help`	Zeigt den Text der Hilfe an.
`--version`	Zeigt die Versionsnummer der Applikation an.
`<command>`	Der auszuführende Befehl. Siehe Liste unten. Jeder Befehl ist in den Unterabschnitten dieses Abschnitts ausführlich mit seinen Optionen und Argumenten beschrieben.
`[options]`	Die Optionen eines Befehls. Diese werden mit ihren jeweiligen Befehlen aufgelistet und im Abschnitt Optionen genauer beschrieben.
`[arguments]`	Das/Die Argument(e) eines Befehls. Diese werden mit ihren jeweiligen Befehlen aufgelistet und beschrieben.

CLI-Befehle

 © 2014 Altova GmbH

Im Folgenden sind die CLI-Befehle nach Funktionalität geordnet aufgelistet. Sie werden in den Unterabschnitten dieses Abschnitts näher beschrieben. (Beachten Sie, dass einige Validierungsbefehle in der Liste unten in mehreren Gruppen vorkommen.)

Validierungsbefehle

valdtd | dtd	Validiert ein DTD-Dokument.
valxml-withdtd | xml	Validiert ein XML-Dokument anhand einer DTD.
valxml-withxsd | xsi	Validiert ein XML-Dokument anhand eines XML-Schemas.
valxquery	Validiert ein XQuery-Dokument.
valxsd | xsd	Validiert ein W3C XML-Schema-Dokument.
valxslt	Validiert ein XSLT-Dokument.
valany	Validiert jedes beliebige Dokument eines Typs, der durch die vorhergehenden Befehle in dieser Liste validiert wird. Der Dokumenttyp wird automatisch erkannt.

Befehle für die Überprüfung der Wohlgeformtheit

wfxml	Überprüft ein XML-Dokument auf Wohlgeformtheit.
wfdtd	Überprüft ein DTD-Dokument auf Wohlgeformtheit.
wfany	Überprüft jedes XML- oder DTD-Dokument auf Wohlgeformtheit.

XSLT-Befehle

xslt	Führt eine Transformation anhand der durch das Argument gelieferten XSLT-Datei durch.
valxslt	Validiert ein XSLT-Dokument.

XQuery-Befehle

xquery	Führt anhand der durch das Argument gelieferten XQuery-Datei eine XQuery durch.
valxquery	Validiert ein XQuery-Dokument.

3.1 XML-, DTD-, XSD-Validierungsbefehle

Die XML-Validierungsbefehle dienen zum Validieren der folgenden Dokumenttypen:

- *XML:* Validiert XML-Instanzdokumente anhand einer DTD (`valxml-withdtd | xml`) oder eines XML-Schemas 1.0/1.1 (`valxml-withxsd | xsi`).
- *DTD:* Überprüft, ob eine DTD wohlgeformt ist und keine Fehler enthält (`valdtd | dtd`).
- *XSD:* Validiert ein W3C XML-Schema-Dokument (XSD) anhand der Regeln in der XML-Schema-Spezifikation (`valxsd | xsd`).

Die XML-Validierungsbefehle sind in den Unterabschnitten dieses Abschnitts näher beschrieben:

`valxml-withdtd | xml`	Validiert ein XML-Instanzdokument anhand einer DTD.
`valxml-withxsd | xsi`	Validiert ein XML-Instanzdokument anhand eines XML-Schemas.
`valdtd | dtd`	Validiert ein DTD-Dokument.
`valxsd | xsd`	Validiert ein W3C XML-Schema-Dokument (XSD).
`valany`	Validiert jedes beliebige XML-, DTD- oder XSD-Dokument. Beachten Sie, dass dieser Befehl auch zum Validieren von XSLT- oder XQuery-Dokumenten verwendet wird. Der Dokumenttyp wird automatisch erkannt.

Anmerkung: Auch XSLT- und XQuery-Dokumente können validiert werden. Diese Validierungsbefehle sind in den entsprechenden Abschnitten XSLT-Befehle und XQuery-Befehle beschrieben.

© 2014 Altova GmbH

3.1.1 valxml-withdtd (xml)

Der Befehl **valxml-withdtd | xml** validiert ein oder mehrere XML-Dokumente anhand einer DTD.

Windows **RaptorXML valxml-withdtd | xml [options]** ***InputFile***
Linux **raptorxml valxml-withdtd | xml [options]** ***InputFile***
Mac **raptorxml valxml-withdtd | xml [options]** ***InputFile***

Das Argument *InputFile* ist das zu validierende XML-Dokument. Wenn das XML-Dokument eine Referenz auf eine DTD enthält, wird die Option --dtd nicht benötigt.

Um mehrere Input-Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (.txt-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf true gesetzten Option --listfile als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

- **raptorxml** valxml-withdtd --dtd=c:\MyDTD.dtd c:\Test.xml
- **raptorxml** xml c:\Test.xml
- **raptorxml** xml --verbose=true c:\Test.xml
- **raptorxml** xml --listfile=true c:\FileList.txt

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung
▼ **dtd**

--dtd = *FILE*
Definiert das für die Validierung zu verwendende externe DTD-Dokument. Wenn das XML-Dokument eine Referenz auf eine externe DTD enthält, setzt die CLI-Option die externe Referenz außer Kraft.

listfile

--listfile = true|false
Bei true wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist false. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die Option --listfile nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

namespaces

--namespaces = true|false
Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

--recurse = true|false
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei true wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: test.zip|zip\test.xml wählt Dateien mit dem Namen test.xml auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter * und ? verwendet werden. Mit *.xml werden folglich alle .xml Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

streaming

--streaming = true|false
Aktiviert die Streaming-Validierung. Standardwert ist true. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie --streaming auf den Wert false setzen). Wenn Sie die Option --script mit dem Befehl valxml-withxsd verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option --streaming ignoriert wird, wenn --parallel-assessment auf true gesetzt wird.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

Kataloge und globale Ressourcen

catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-

Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

- **user-catalog**

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

- **enable-globalresources**

  --enable-globalresources = `true|false`
  Aktiviert die globalen Ressourcen. Standardwert ist `false`.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- **globalresourceconfig [gc]**

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

- **globalresourcefile [gr]**

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - **log-output**

    --log-output = *FILE*

Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

 © 2014 Altova GmbH

3.1.2 valxml-withxsd (xsi)

Der Befehl **valxml-withxsd | xsi** validiert ein oder mehrere XML-Instanzdokumente anhand der W3C XML Schema Definition Language (XSD) Spezifikationen 1.0 und 1.1.

Windows	**RaptorXML valxml-withxsd | xsi [options] *InputFile***
Linux	**raptorxml valxml-withxsd | xsi [options] *InputFile***
Mac	**raptorxml valxml-withxsd | xsi [options] *InputFile***

Das Argument *InputFile* ist das zu validierende XML-Dokument. --schemalocation-hints=true|false gibt an, ob die XSD-Referenz im XML-Dokument verwendet werden soll oder nicht, wobei der Standardwert true ist (der Pfad wird verwendet). Mit der Option --xsd=FILE wird das/die zu verwendende(n) Schema(s) angegeben.

Um mehrere Input-Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (.txt-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf true gesetzten Option --listfile als das *InputFile* Argument an (*siehe Optionsliste unten*).

Anmerkung: Wenn Sie zum Ausführen von Python-Skripts die Option --script verwenden, stellen Sie sicher, dass Sie auch --streaming=false definieren.

Beispiele

- **raptorxml** valxml-withxsd --schemalocation-hints=false --xsd=c:\MyXSD.xsd c:\HasNoXSDRef.xml
- **raptorxml** xsi c:\HasXSDRef.xml
- **raptorxml** xsi --xsd-version=1.1 --listfile=true c:\FileList.txt

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung ung Verarbeitung
▼ **assessment-mode**

--assessment-mode = lax|strict
Definiert den Beurteilungsmodus für die Gültigkeit von Schemas gemäß der XSD-Spezifikation. Der Standardwert ist strict. Das XML-Instanzdokument wird entsprechend dem mit dieser Option definierten Modus validiert.

listfile

--listfile = true|false
Bei true wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist false. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die Option --listfile nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

parallel-assessment [pa]

--pa | --parallel-assessment = true|false
Bei Setzung auf true wird die Schema-Validierung parallel ausgeführt. Das bedeutet, wenn sich auf irgendeiner Ebene mehr als 128 Elemente befinden, so werden diese Elemente über mehrere Threads parallel verarbeitet. Auf diese Weise können besonders große XML-Dateien schneller verarbeitet werden, wenn diese Option aktiv ist. Parallele Validierungen können gleichzeitig auf einer hierarchischen Ebene ausgeführt werden, können in einem einzigen Infoset aber auch auf mehreren Ebenen erfolgen. Beachten Sie dass die parallele Validierung im Streaming-Modus nicht funktioniert. Aus diesem Grund wird die Option --streaming ignoriert, wenn --parallel-assessment auf true. gesetzt ist. Außerdem wird bei Verwendung der Option --parallel-assessment mehr Arbeitsspeicher benötigt. Die Standardeinstellung ist false. Die Kurzform für die Option ist --pa.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

--recurse = true|false
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei true wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: test.zip|zip\test.xml wählt Dateien mit dem Namen test.xml auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter * und ? verwendet werden. Mit *.xml werden folglich alle .xml Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Definiert das Verhalten von xs:import Elementen, von denen jedes ein optionales namespace Attribut und ein optionales schemaLocation Attribut hat: <import namespace="someNS" schemaLocation="someURL">. Sie verhalten sich folgendermaßen:

- load-by-schemalocation: Anhand des Werts des schemaLocation Attributs wird

der Schemapfad ermittelt, wobei Katalog-Mappings berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).

- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping.
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping hat, so wird das Katalog-Mapping verwendet. Wenn beide Katalog-Mappings haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping). Der Standardwert ist `prefer-schemalocation`.

script

`--script =` *`FILE`*

Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus.

script-param

`--script-param =` *`KEY:VALUE`*

Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Scripts aufgerufen werden können.

streaming

--streaming = true|false
Aktiviert die Streaming-Validierung. Standardwert ist true. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie --streaming auf den Wert false setzen). Wenn Sie die Option --script mit dem Befehl valxml-withxsd verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option --streaming ignoriert wird, wenn --parallel-assessment auf true gesetzt wird.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

xinclude

--xinclude = true|false
Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist false. Bei false werden die include-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

xml-mode

--xml-mode = wf|id|valid
Definiert den zu verwendenden XML-Verarbeitungsmodus: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf.

xsd

--xsd = *FILE*
Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehrere Schema-Dokumente zu definieren, fügen Sie die Option mehrmals hinzu.

xsd-version

--xsd-version = 1.0|1.1|detect
Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist 1.0. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option detect ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (1.0 oder 1.1) durch Lesen des Werts des vc:minVersion Attributs des <xs:schema> Elements des Dokuments ermittelt werden. Wenn der Wert des @vc:minVersion Attributs 1.1 ist, wird das Schema als Version 1.1 erkannt. Bei jedem anderen Wert wird das Schema als 1.0 erkannt. Bei jedem anderen Wert oder bei Fehlen des @vc:minVersion Attributs wird das Schema als Version 1.0 gelesen.

Kataloge und globale Ressourcen

catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (<installationsordner>\Altova\RaptorXMLServer2015\etc

\RootCatalog.xml). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

- **user-catalog**

  --user-catalog = *FILE*
  Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

- **enable-globalresources**

  --enable-globalresources = true|false
  Aktiviert die globalen Ressourcen. Standardwert ist false.
  Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

- **globalresourceconfig [gc]**

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

- **globalresourcefile [gr]**

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = text|shortxml|longxml
    Definiert das Format der Fehlerausgabe. Der Standardwert ist text. Mit den anderen Optionen werden XML-Formate generiert, wobei mit longxml mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist 100. Zulässig sind Werte von 1 bis 999. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: valany --h. (Alternativ dazu kann der Befehl help zusammen mit einem Argument verwendet werden. Beispiel: help valany.)

  - **log-output**

    --log-output = *FILE*
    Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie

sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

 © 2014 Altova GmbH

3.1.3 valdtd (dtd)

Der Befehl **valdtd | dtd** validiert ein oder mehrere DTD-Dokumente anhand der XML 1.0 oder 1.1-Spezifikation.

Windows	**RaptorXML valdtd | dtd [options]** ***InputFile***
Linux	**raptorxml valdtd | dtd [options]** ***InputFile***
Mac	**raptorxml valdtd | dtd [options]** ***InputFile***

Das Argument *InputFile* ist das zu validierende DTD-Dokument. Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

- `raptorxml valdtd c:\Test.dtd`
- **raptorxml** `dtd --verbose=true c:\Test.dtd`
- **raptorxml** `dtd --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung

▼ **listfile**

--listfile = `true|false`
Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die

Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

`--recurse =` `true|false`
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `test.zip|zip\test.xml` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Kataloge und globale Ressourcen

catalog

`--catalog =` *`FILE`*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

`--user-catalog =` *`FILE`*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

`--enable-globalresources =` `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

globalresourceconfig [gc]

`--gc | --globalresourceconfig =` *`VALUE`*
Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

globalresourcefile [gr]

`--gr | --globalresourcefile =` *`FILE`*
Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

Meldungen, Fehler, Hilfe, Timeout, Version

 © 2014 Altova GmbH

error-format

--error-format = `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N*
Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.1.4 valxsd (xsd)

Der Befehl **valxsd | xsd** validiert ein oder mehrere XML-Schema-Dokumente (XSD-Dokumente) anhand der W3C XML Schema Definition Language (XSD) Spezifikationen 1.0 und 1.1. Beachten Sie, dass das Schema selbst anhand der XML-Schema-Spezifikation validiert wird und nicht ein XML-Instanzdokument.

Windows **RaptorXML valxsd | xsd [options]** ***InputFile***
Linux **raptorxml valxsd | xsd [options]** ***InputFile***
Mac **raptorxml valxsd | xsd [options]** ***InputFile***

Das Argument *InputFile* ist das zu validierende XML-Schema-Dokument. Die Option --xsd-version=1.0|1.1|detect gibt an, anhand welcher XSD-Version das Dokument validiert werden soll, wobei der Standardwert 1.0 ist.

Um mehrere Dokumente zu validieren, (i) listen Sie entweder die zu validierenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu validierenden Dateien in einer Textdatei (.txt-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf true gesetzten Option --listfile als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

- **raptorxml** valxsd c:\Test.xsd
- **raptorxml** xsd --verbose=true c:\Test.xsd
- **raptorxml** xsd --listfile=true c:\FileList.txt

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung
▼ **listfile**

--listfile = true|false
Bei true wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist false. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie

allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

`--recurse =` `true|false`
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `test.zip|zip\test.xml` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

schema-imports

`--schema-imports =` `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Sie verhalten sich folgendermaßen:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

schema-location-hints

`--schema-location-hints =` `load-by-schemalocation | load-by-namespace | load-combining-both | ignore`
- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping.
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping hat, so wird das Katalog-Mapping verwendet.

Wenn beide Katalog-Mappings haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.

- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

schema-mapping

--schema-mapping = `prefer-schemalocation | prefer-namespace`

Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping). Der Standardwert ist `prefer-schemalocation`.

script

--script = *FILE*

Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus.

script-param

--script-param = *KEY:VALUE*

Zusätzliche benutzerdefinierte Parameter, die während der Ausführung von Python Scripts aufgerufen werden können.

xinclude

--xinclude = `true|false`

Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

xml-mode

--xml-mode = `wf|id|valid`

Definiert den zu verwendenden XML-Verarbeitungsmodus: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`.

xsd-version

--xsd-version = `1.0|1.1|detect`

Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

 © 2014 Altova GmbH

- Kataloge und globale Ressourcen
  - **catalog**

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **user-catalog**

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **enable-globalresources**

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **globalresourceconfig [gc]**

    --gc | --globalresourceconfig = *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

  - **globalresourcefile [gr]**

    --gr | --globalresourcefile = *FILE*
    Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl

help zusammen mit einem Argument verwendet werden. Beispiel: help valany.)

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: 40.

verbose

--verbose = true|false
Mit dem Wert true wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie --version bei Verwendung mit einem Befehl vor den Befehl.

 © 2014 Altova GmbH

3.1.5 valany

Der Befehl **valany** validiert ein XML-, DTD- oder XML-Schema-Dokument anhand der jeweiligen Spezifikation(en). Der Typ des Dokuments wird automatisch erkannt.

Windows **RaptorXML valany [options]** ***InputFile***
Linux **raptorxml valany [options]** ***InputFile***
Mac **raptorxml valany [options]** ***InputFile***

Das Argument *InputFile* gibt das zu validierende Dokument an. Beachten Sie, dass nur ein Dokument als Argument des Befehls angegeben werden kann. Der Typ des angegebenen Dokuments wird automatisch erkannt.

Beispiele

- raptorxml valany c:\Test.xml
- **raptorxml** valany --errorformat=text c:\Test.xml

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung

▼ **recurse**

--recurse = true|false
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei true wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: test.zip|zip\test.xml wählt Dateien mit dem Namen test.xml auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter * und ? verwendet werden. Mit *.xml werden folglich alle .xml Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ **schema-imports**

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only

Definiert das Verhalten von xs:import Elementen, von denen jedes ein optionales namespace Attribut und ein optionales schemaLocation Attribut hat: <import namespace="someNS" schemaLocation="someURL">. Sie verhalten sich folgendermaßen:

- load-by-schemalocation: Anhand des Werts des schemaLocation Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- load-preferring-schemalocation: Wenn das schemaLocation Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings berücksichtigt werden. Falls kein schemaLocation Attribut vorhanden ist, wird der Wert des namespace Attributs über ein Katalog-Mapping verwendet. Dies ist der **Standardwert**.
- load-by-namespace: Anhand des Werts des namespace Attributs wird der Schemapfad über ein Katalog-Mapping ermittelt.
- load-combining-both: Wenn entweder das Attribut namespace oder das Attribut schemaLocation ein Katalog-Mapping hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings haben, ist es vom Wert der Option --schema-mapping (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping vorhanden ist, wird das schemaLocation Attribut verwendet.
- license-namespace-only: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

schema-location-hints

--schema-location-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore

- Der Wert load-by-schemalocation verwendet die URL des Schemapfads in den Attributen xsi:schemaLocation und xsi:noNamespaceSchemaLocation in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert load-by-namespace verwendet den Namespace-Teil von xsi:schemaLocation und - im Fall von xsi:noNamespaceSchemaLocation - einen leeren String und ermittelt das Schema über ein Katalog-Mapping.
- Bei Verwendung von load-combining-both und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping hat, so wird das Katalog-Mapping verwendet. Wenn beide Katalog-Mappings haben, ist es vom Wert der --schema-mapping Option (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option ignore ist, werden die beiden Attribute xsi:schemaLocation und xsi:noNamespaceSchemaLocation ignoriert.

schema-mapping

--schema-mapping = prefer-schemalocation | prefer-namespace

Wenn eine der Optionen --schemalocation-hints oder --schema-imports einen Wert load-combining-both hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert prefer-schemalocation bezieht sich auf das URL-Mapping). Der Standardwert ist prefer-schemalocation.

Kataloge und globale Ressourcen

catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

Meldungen, Fehler, Hilfe, Timeout, Version

error-format

--error-format = `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N*
Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

 © 2014 Altova GmbH

3.2 Befehle für die Überprüfung der Wohlgeformtheit

Mit Hilfe der Befehle zur Überprüfung der Wohlgeformtheit kann die Wohlgeformtheit von XML-Dokumenten und DTDs überprüft werden. Diese Befehle sind unten aufgelistet und in den Unterabschnitten dieses Abschnitts näher beschrieben:

wfxml	Überprüft ein XML-Dokument auf Wohlgeformtheit.
wfdtd	Überprüft ein DTD-Dokument auf Wohlgeformtheit.
wfany	Überprüft jedes XML- oder DTD-Dokument auf Wohlgeformtheit. Der Typ des Dokuments wird automatisch erkannt.

3.2.1 wfxml

Der Befehl **wfxml** überprüft ein oder mehrere XML-Dokumente anhand der XML 1.0- oder XML 1.1-Spezifikation auf Wohlgeformtheit.

Windows **RaptorXML wfxml [options]** ***InputFile***
Linux **raptorxml wfxml [options]** ***InputFile***
Mac **raptorxml wfxml [options]** ***InputFile***

Das Argument *InputFile* ist das XML-Dokument, das auf Wohlgeformtheit überprüft werden soll. Um mehrere Input-Dokumente auf Wohlgeformtheit zu prüfen, (i) listen Sie entweder die zu überprüfenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu überprüfenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

- `raptorxml wfxml c:\Test.xml`
- **`raptorxml`** `wfxml --verbose=true c:\Test.xml`
- **`raptorxml`** `wfxml --listfile=true c:\FileList.txt`

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung

▼ **dtd**

--dtd = *FILE*
Definiert das für die Validierung zu verwendende externe DTD-Dokument. Wenn das XML-Dokument eine Referenz auf eine externe DTD enthält, setzt die CLI-Option die externe Referenz außer Kraft.

▼ **listfile**

 © 2014 Altova GmbH

--listfile = `true|false`
Bei `true` wird das Argument *`InputFile`* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

namespaces

--namespaces = `true|false`
Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

--recurse = `true|false`
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `test.zip|zip\test.xml` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Kataloge und globale Ressourcen

catalog

--catalog = *`FILE`*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

--user-catalog = *`FILE`*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- **globalresourceconfig [gc]**

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

- **globalresourcefile [gr]**

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - **log-output**

    --log-output = *FILE*
    Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - **network-timeout**

    --network-timeout = *VALUE*
    Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

  - **verbose**

    --verbose = `true|false`
    Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **verbose-output**

 © 2014 Altova GmbH

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.2.2 wfdtd

Der Befehl **wfdtd** überprüft ein oder mehrere DTD-Dokumente anhand der XML 1.0- oder XML 1.1-Spezifikation auf Wohlgeformtheit.

Windows **RaptorXML wfdtd [options]** ***InputFile***
Linux **raptorxml wfdtd [options]** ***InputFile***
Mac **raptorxml wfdtd [options]** ***InputFile***

Das Argument *InputFile* ist das zu überprüfende DTD-Dokument. Um mehrere Dokumente auf Wohlgeformtheit zu prüfen, (i) listen Sie entweder die zu überprüfenden Dateien im CLI auf, wobei jede Datei durch ein Leerzeichen von der nächsten getrennt wird, oder (ii) listen Sie die zu überprüfenden Dateien in einer Textdatei (`.txt`-Datei) auf (ein Dateiname pro Zeile) und geben Sie diese Textdatei zusammen mit der auf `true` gesetzten Option `--listfile` als das *InputFile* Argument an (*siehe Optionsliste unten*).

Beispiele

- **raptorxml** wfdtd c:\Test.dtd
- **raptorxml** wfdtd --verbose=true c:\Test.dtd
- **raptorxml** wfdtd --listfile=true c:\FileList.txt

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung
▼ **listfile**

--listfile = `true|false`
Bei `true` wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist `false`. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie

allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die Option `--listfile` nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

recurse

`--recurse =` `true|false`
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei `true` wählt das Argument *`InputFile`* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `test.zip|zip\test.xml` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Kataloge und globale Ressourcen

catalog

`--catalog =` *`FILE`*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

`--user-catalog =` *`FILE`*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

`--enable-globalresources =` `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

globalresourceconfig [gc]

`--gc | --globalresourceconfig =` *`VALUE`*
Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

globalresourcefile [gr]

`--gr | --globalresourcefile =` *`FILE`*
Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

    --help
    Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

  - **log-output**

    --log-output = *FILE*
    Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

  - **network-timeout**

    --network-timeout = *VALUE*
    Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

  - **verbose**

    --verbose = `true|false`
    Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **verbose-output**

    --verbose-output = *FILE*
    Schreibt die ausführliche Ausgabe in *FILE*.

  - **version**

    --version
    Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

 © 2014 Altova GmbH

3.2.3 wfany

Der Befehl **valany** überprüft die Wohlgeformtheit eines XML-, DTD- oder XML-Schema-Dokuments anhand der jeweiligen Spezifikation(en). Der Typ des Dokuments wird automatisch erkannt.

Windows **RaptorXML wfany [options]** ***InputFile***
Linux **raptorxml wfany [options]** ***InputFile***
Mac **raptorxml wfany [options]** ***InputFile***

Das Argument *InputFile* gibt das zu validierende Dokument an. Beachten Sie, dass nur ein Dokument als Argument des Befehls angegeben werden kann. Der Typ des angegebenen Dokuments wird automatisch erkannt.

Beispiele

- `raptorxml wfany c:\Test.xml`
- **raptorxml** `wfany --errorformat=text c:\Test.xml`

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ Validierung und Verarbeitung
▼ **recurse**

--recurse = `true|false`
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `test.zip|zip\test.xml` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Kataloge und globale Ressourcen
  - **catalog**

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **user-catalog**

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **enable-globalresources**

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **globalresourceconfig [gc]**

    --gc | --globalresourceconfig = *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

  - **globalresourcefile [gr]**

    --gr | --globalresourcefile = *FILE*
    Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

© 2014 Altova GmbH

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.3 XSLT-Befehle

Die XSLT-Befehle sind:

- xslt: zum Transformieren von XML-Dokumenten anhand eines XSLT-Dokuments
- valxslt: zum Validieren von XSLT-Dokumenten

Die Argumente und Optionen für die einzelnen Befehle sind in den Unterabschnitten xslt und valxslt aufgelistet.

© 2014 Altova GmbH

3.3.1 xslt

Der Befehl **xslt** erhält als einziges Argument eine XSLT-Datei und transformiert anhand dieser Datei eine XML-Input-Datei in eine Ausgabedatei. Die Input-Datei und die Ausgabedatei sind als Optionen definiert.

Windows **RaptorXML xslt [options]** ***XSLT-File***
Linux **raptorxml xslt [options]** ***XSLT-File***
Mac **raptorxml xslt [options]** ***XSLT-File***

Das Argument *XSLT-File* ist der Pfad und Name der für die Transformation zu verwendenden XSLT-Datei. Es wird eine XML-Input-Datei (--input) oder eine benannte Vorlage als Eintrittspunkt (--template-entry-point) benötigt. Wenn keine --output Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben. Sie können XSLT 1.0, 2.0 oder 3.0 verwenden. Standardmäßig wird XSLT 3.0 verwendet.

Beispiele

- **raptorxml** xslt --input=c:\Test.xml --output=c:\Output.xml c:\Test.xslt
- **raptorxml** xslt --template-entry-point=StartTemplate --output=c:\Output.xml c:\Test.xslt
- **raptorxml** xslt --input=c:\Test.xml --output=c:\Output.xml --param=date://node[1]/@att1 --p=title:'stringwithoutspace' --param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ XSLT-Verarbeitung

▼ **indent-characters**

--indent-characters = *VALUE*
Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

▼ **input**

--input = *FILE*
Die URL der zu transformierenden XML-Datei.

output

output = *FILE*
Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Wenn keine `--output` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

param [p]

--p | --param = *KEY:VALUE*

XQuery
Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

XSLT
Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

streaming

--streaming = `true|false`
Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.

 © 2014 Altova GmbH

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

template-entry-point

--template-entry-point = *VALUE*
Gibt den Namen einer benannten Vorlage im XSLT-Stylesheet an, das der Eintrittspunkt der Transformation ist.

template-mode

--template-mode = *VALUE*
Definiert den Vorlagenmodus für die Transformation.

xslt-version

--xslt-version = `1|2|3`
Definiert, ob der XSLT-Prozessor XSLT 1.0, XSLT 2.0 oder XSLT 3.0 verwenden soll. Der Standardwert ist `3`.

XML-Schema und XML-Instanz

load-xml-with-psvi

--load-xml-with-psvi = `true|false`
Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `false`.

schema-imports

--schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Sie verhalten sich folgendermaßen:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

schema-location-hints

--schema-location-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore

- Der Wert load-by-schemalocation verwendet die URL des Schemapfads in den Attributen xsi:schemaLocation und xsi:noNamespaceSchemaLocation in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert load-by-namespace verwendet den Namespace-Teil von xsi:schemaLocation und - im Fall von xsi:noNamespaceSchemaLocation - einen leeren String und ermittelt das Schema über ein Katalog-Mapping.
- Bei Verwendung von load-combining-both und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping hat, so wird das Katalog-Mapping verwendet. Wenn beide Katalog-Mappings haben, ist es vom Wert der --schema-mapping Option (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option ignore ist, werden die beiden Attribute xsi:schemaLocation und xsi:noNamespaceSchemaLocation ignoriert.

schema-mapping

--schema-mapping = prefer-schemalocation | prefer-namespace

Wenn eine der Optionen --schemalocation-hints oder --schema-imports einen Wert load-combining-both hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert prefer-schemalocation bezieht sich auf das URL-Mapping). Der Standardwert ist prefer-schemalocation.

xinclude

--xinclude = true|false

Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist false. Bei false werden die include-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

xml-mode

--xml-mode = wf|id|valid

Definiert den zu verwendenden XML-Verarbeitungsmodus: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf.

xsd-version

--xsd-version = 1.0|1.1|detect

Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist 1.0. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option detect ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (1.0 oder 1.1) durch Lesen des Werts des vc:minVersion Attributs des <xs:schema> Elements des Dokuments ermittelt werden. Wenn der Wert des @vc:minVersion Attributs 1.1 ist, wird das Schema als Version 1.1 erkannt. Bei jedem anderen Wert wird das Schema als 1.0 erkannt. Bei jedem anderen Wert oder bei Fehlen des @vc:minVersion Attributs wird das Schema als Version 1.0 gelesen.

Kataloge und globale Ressourcen

 © 2014 Altova GmbH

catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

Erweiterungen
Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

chartext-disable

--chartext-disable = `true|false`
Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

dotnetext-disable

--dotnetext-disable = `true|false`
Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

javaext-barcode-location

--javaext-barcode-location = *FILE*
Definiert den Pfad der Barcode-Erweiterungsdatei.

javaext-disable

--javaext-disable = `true|false`
Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Meldungen, Fehler, Hilfe, Timeout, Version

error-format

--error-format = `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N*
Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit

 © 2014 Altova GmbH

einem Befehl vor den Befehl.

3.3.2 valxslt

Der Befehl **valxslt** erhält als einziges Argument eine XSLT-Datei und validiert diese.

Windows **RaptorXML valxslt [options]** ***XSLT-File***
Linux **raptorxml valxslt [options]** ***XSLT-File***
Mac **raptorxml valxslt [options]** ***XSLT-File***

Das Argument *XSLT-File* ist der Pfad und Name der zu validierenden XSLT-Datei. Die Validierung kann anhand der XSLT 1.0, 2.0 oder 3.0-Spezifikation erfolgen. Standardmäßig wird die XSLT 3.0-Spezifikation verwendet.

Beispiele

- `raptorxml valxslt c:\Test.xslt`
- **raptorxml** `valxslt --xslt-version=2 c:\Test.xslt`

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ XSLT-Verarbeitung

▼ **chartext-disable**

--chartext-disable = `true|false`
Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ **dotnetext-disable**

--dotnetext-disable = `true|false`
Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ **javaext-barcode-location**

 © 2014 Altova GmbH

--javaext-barcode-location = *FILE*
Definiert den Pfad der Barcode-Erweiterungsdatei.

javaext-disable

--javaext-disable = `true|false`
Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

template-entry-point

--template-entry-point = *VALUE*
Gibt den Namen einer benannten Vorlage im XSLT-Stylesheet an, das der Eintrittspunkt der Transformation ist.

template-mode

--template-mode = *VALUE*
Definiert den Vorlagenmodus für die Transformation.

xslt-version

--xslt-version = `1|2|3`
Definiert, ob der XSLT-Prozessor XSLT 1.0, XSLT 2.0 oder XSLT 3.0 verwenden soll. Der Standardwert ist `3`.

XML-Schema und XML-Instanz

schema-imports

--schema-imports = `load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only`
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Sie verhalten sich folgendermaßen:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

schema-location-hints

--schema-location-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore

- Der Wert load-by-schemalocation verwendet die URL des Schemapfads in den Attributen xsi:schemaLocation und xsi:noNamespaceSchemaLocation in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert load-by-namespace verwendet den Namespace-Teil von xsi:schemaLocation und - im Fall von xsi:noNamespaceSchemaLocation - einen leeren String und ermittelt das Schema über ein Katalog-Mapping.
- Bei Verwendung von load-combining-both und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping hat, so wird das Katalog-Mapping verwendet. Wenn beide Katalog-Mappings haben, ist es vom Wert der --schema-mapping Option (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.
- Wenn der Wert der Option ignore ist, werden die beiden Attribute xsi:schemaLocation und xsi:noNamespaceSchemaLocation ignoriert.

schema-mapping

--schema-mapping = prefer-schemalocation | prefer-namespace

Wenn eine der Optionen --schemalocation-hints oder --schema-imports einen Wert load-combining-both hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert prefer-schemalocation bezieht sich auf das URL-Mapping). Der Standardwert ist prefer-schemalocation.

xinclude

--xinclude = true|false

Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist false. Bei false werden die include-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

xml-mode

--xml-mode = wf|id|valid

Definiert den zu verwendenden XML-Verarbeitungsmodus: wf=Wohlgeformtheitsprüfung; id=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; valid=Validierung. Der Standardwert ist wf.

xsd-version

--xsd-version = 1.0|1.1|detect

Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist 1.0. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option detect ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (1.0 oder 1.1) durch Lesen des Werts des vc:minVersion Attributs des <xs:schema> Elements des Dokuments ermittelt werden. Wenn der Wert des @vc:minVersion Attributs 1.1 ist, wird das Schema als Version 1.1 erkannt. Bei jedem anderen Wert wird das Schema als 1.0 erkannt. Bei jedem anderen Wert oder bei Fehlen des @vc:minVersion Attributs wird das Schema als Version 1.0 gelesen.

Kataloge und globale Ressourcen

 © 2014 Altova GmbH

catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

Erweiterungen
Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.

chartext-disable

--chartext-disable = `true|false`
Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

dotnetext-disable

--dotnetext-disable = `true|false`
Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

javaext-barcode-location

--javaext-barcode-location = *FILE*
Definiert den Pfad der Barcode-Erweiterungsdatei.

javaext-disable

--javaext-disable = true|false
Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

Meldungen, Fehler, Hilfe, Timeout, Version

error-format

--error-format = text|shortxml|longxml
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N*
Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = true|false
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit

 © 2014 Altova GmbH

einem Befehl vor den Befehl.

3.4 XQuery-Befehle

Die XQuery-Befehle sind:

- xquery: zum Ausführen von XQuery-Dokumenten, optional mit einem Input-Dokument
- xqueryupdate: zum Ausführen eines XQuery Update über ein XQuery-Dokument und optional mit einem zu aktualisierenden Input-Dokument
- valxquery: zum Validieren von XQuery-Dokumenten

Die Argumente und Optionen für die einzelnen Befehle sind in den Unterabschnitten xquery und valxquery aufgelistet.

© 2014 Altova GmbH

3.4.1 xquery

Der Befehl **xquery** erhält als einziges Argument eine XQuery-Datei und führt diese mit einer optionalen Input-Datei aus, um eine Ausgabedatei zu erzeugen. Die Input-Datei und die Ausgabedatei sind als Optionen definiert.

Windows **RaptorXML xquery [options]** ***XQuery-File***
Linux **raptorxml xquery [options]** ***XQuery-File***
Mac **raptorxml xquery [options]** ***XQuery-File***

Das Argument ***XQuery-File*** ist der Pfad und Name der auszuführenden XQuery-Datei. Sie können XQuery 1.0 oder 3.0 verwenden. Standardmäßig wird XQuery 3.0 verwendet.

Beispiele

- **raptorxml** xquery --output=c:\Output.xml c:\TestQuery.xq
- **raptorxml** xquery --input=c:\Input.xml --output=c:\Output.xml --param=company:"Altova" --p=date:"2006-01-01" c:\TestQuery.xq
- **raptorxml** xquery --input=c:\Input.xml --output=c:\Output.xml --param=source:" doc('c:\test\books.xml')//book "
- **raptorxml** xquery --output=c:\Output.xml --omit-xml-declaration=false --output-encoding=ASCII c:\TestQuery.xq

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ XQuery-Verarbeitung

▼ **indent-characters**

--indent-characters = *VALUE*
Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

▼ **input**

--input = *FILE*
Die URL der zu transformierenden XML-Datei.

omit-xml-declaration

--omit-xml-declaration = `true|false`

Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

output

output = *FILE*

Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Wenn keine `--output` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

output-encoding

--output-encoding = *VALUE*

Der Wert des Kodierungsattributs im Ausgabedokument. Gültige Werte sind die Namen im IANA-Zeichensatz-Register. Der Standardwert ist `UTF-8`.

output-indent

--output-indent = `true|false`

Wenn der Wert `true` ist, wird die Ausgabe entsprechend ihrer hierarchischen Struktur eingerückt. Bei `false` gibt es keine hierarchische Einrückung. Der Standardwert ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

output-method

--output-method = `xml|html|xhtml|text`

Definiert das Ausgabeformat. Der Standardwert ist `xml`.

param [p]

--p | --param = *KEY:VALUE*

XQuery

Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält,

 © 2014 Altova GmbH

muss er in doppelte Anführungszeichen gesetzt werden.

- ***XSLT***

  Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

  ```
  raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
  param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
  param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
  ```

- **xquery-version**

  --xquery-version = `1|3`

  Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist `1`.

- XML-Schema und XML-Instanz
  - **load-xml-with-psvi**

    --load-xml-with-psvi = `true|false`

    Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `false`.

  - **xinclude**

    --xinclude = `true|false`

    Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **xml-mode**

    --xml-mode = `wf|id|valid`

    Definiert den zu verwendenden XML-Verarbeitungsmodus: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`.

  - **xsd-version**

    --xsd-version = `1.0|1.1|detect`

    Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - **catalog**

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **user-catalog**

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **enable-globalresources**

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen. Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **globalresourceconfig [gc]**

    --gc | --globalresourceconfig = *VALUE*
    Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

  - **globalresourcefile [gr]**

    --gr | --globalresourcefile = *FILE*
    Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

  - **error-limit**

    --error-limit = *N*
    Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

  - **help**

    --help

 © 2014 Altova GmbH

Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.4.2 valxquery

Der Befehl **valxquery** erhält als einziges Argument eine XQuery-Datei und validiert diese.

Windows **RaptorXML valxquery [options] *XQuery-File***
Linux **raptorxml valxquery [options] *XQuery-File***
Mac **raptorxml valxquery [options] *XQuery-File***

Das Argument ***XQuery-File*** ist der Pfad und Name der zu validierenden XQuery-Datei.

Beispiele

- `raptorxml valxquery c:\Test.xquery`
- **raptorxml** `valxquery --xquery-version=1 c:\Test.xquery`

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ XQuery-Verarbeitung

▼ **omit-xml-declaration**

--omit-xml-declaration = `true|false`
Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ **xquery-version**

--xquery-version = `1|3`
Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist `1`.

▼ XML-Schema und XML-Instanz

 © 2014 Altova GmbH

- **load-xml-with-psvi**

  --load-xml-with-psvi = `true|false`
  Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `false`.

- **xinclude**

  --xinclude = `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- **xml-mode**

  --xml-mode = `wf|id|valid`
  Definiert den zu verwendenden XML-Verarbeitungsmodus: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`.

- **xsd-version**

  --xsd-version = `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - **catalog**

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **user-catalog**

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **enable-globalresources**

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen. Standardwert ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- **globalresourceconfig [gc]**

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

- **globalresourcefile [gr]**

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Erweiterungen
  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.
  - **chartext-disable**

    --chartext-disable = `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **dotnetext-disable**

    --dotnetext-disable = `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **javaext-barcode-location**

    --javaext-barcode-location = *FILE*
    Definiert den Pfad der Barcode-Erweiterungsdatei.

  - **javaext-disable**

    --javaext-disable = `true|false`
    Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

 © 2014 Altova GmbH

error-limit

--error-limit = *N*
Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.4.3 valxqueryupdate

Der Befehl **valxqueryupdate** erhält als einziges Argument eine XQuery-Datei und validiert diese.

Windows **RaptorXML valxquery [options] *XQuery-File***
Linux **raptorxml valxquery [options] *XQuery-File***
Mac **raptorxml valxquery [options] *XQuery-File***

Das Argument ***XQuery-File*** ist der Pfad und Name der zu validierenden XQuery-Datei.

Beispiele

- **raptorxml** valxqueryupdae c:\Test.xqu
- **raptorxml** valxqueryupdate --xquery-version=1 c:\Test.xqu

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

Optionen
Die Optionen des Befehls sind unterhalb nach Gruppen geordnet aufgelistet. Die Werte aller Optionen können ohne Anführungszeichen angegeben werden. Eine Ausnahme bilden die beiden folgenden Fälle: (i) wenn der Wertstring Leerzeichen enthält oder (ii) wenn in der Beschreibung der Option ausdrücklich angegeben ist, dass Anführungszeichen benötigt werden.

▼ XQuery-Verarbeitung

▼ **omit-xml-declaration**

--omit-xml-declaration = true|false
Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert true ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert false ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

▼ **xquery-version**

--xquery-version = 1|3
Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist 1.

▼ XML-Schema und XML-Instanz

 © 2014 Altova GmbH

- **load-xml-with-psvi**

  --load-xml-with-psvi = `true|false`
  Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `false`.

- **xinclude**

  --xinclude = `true|false`
  Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
  Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- **xml-mode**

  --xml-mode = `wf|id|valid`
  Definiert den zu verwendenden XML-Verarbeitungsmodus: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`.

- **xsd-version**

  --xsd-version = `1.0|1.1|detect`
  Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

- Kataloge und globale Ressourcen
  - **catalog**

    --catalog = *FILE*
    Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`). Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **user-catalog**

    --user-catalog = *FILE*
    Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

  - **enable-globalresources**

    --enable-globalresources = `true|false`
    Aktiviert die globalen Ressourcen. Standardwert ist `false`.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- **globalresourceconfig [gc]**

  --gc | --globalresourceconfig = *VALUE*
  Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

- **globalresourcefile [gr]**

  --gr | --globalresourcefile = *FILE*
  Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

- Erweiterungen
  Diese Optionen definieren die Behandlung von speziellen Erweiterungsfunktionen, die in einer Reihe von Enterprise Versionen von Altova-Produkten (wie z.B. in XMLSpy Enterprise Edition) verfügbar sind. Die Verwendung dieser Funktionen ist im Benutzerhandbuch des jeweiligen Produkts beschrieben.
  - **chartext-disable**

    --chartext-disable = `true|false`
    Deaktiviert Diagrammerweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **dotnetext-disable**

    --dotnetext-disable = `true|false`
    Deaktiviert .NET-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

  - **javaext-barcode-location**

    --javaext-barcode-location = *FILE*
    Definiert den Pfad der Barcode-Erweiterungsdatei.

  - **javaext-disable**

    --javaext-disable = `true|false`
    Deaktiviert Java-Erweiterungen. Der Standardwert ist `false`.
    Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

- Meldungen, Fehler, Hilfe, Timeout, Version
  - **error-format**

    --error-format = `text|shortxml|longxml`
    Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

 © 2014 Altova GmbH

error-limit

--error-limit = *N*

Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help

Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

network-timeout

--network-timeout = *VALUE*

Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`

Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*

Schreibt die ausführliche Ausgabe in *FILE*.

version

--version

Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.5 Die Befehle zur Hilfe und Lizenzierung

In diesem Abschnitt sind zwei wichtige Features von RaptorXML Server beschrieben:

- Hilfebefehl: Hier wird beschrieben, wie Sie Informationen zu den verfügbaren Befehlen oder den Argumenten und Optionen eines Befehls anzeigen.
- Lizenzbefehle: Hier wird beschrieben, wie Sie RaptorXML lizenzieren.

© 2014 Altova GmbH

3.5.1 Hilfebefehl

Der Befehl **help** erhält ein einziges Argument: den Namen des Befehls, zu dem die Hilfe aufgerufen werden soll. Er zeigt die Syntax des Befehls sowie andere für die korrekte Ausführung des Befehls wichtigen Informationen an.

Windows **RaptorXML help *Command***
Linux **raptorxml help *Command***
Mac **raptorxml help *Command***

Anmerkung: Wenn kein Argument angegeben wird, werden bei Ausführung des Befehls `help` alle verfügbaren Befehle angezeigt, wobei zu jedem dieser Befehle eine kurze Beschreibung angezeigt wird.

Beispiel
Beispiel für den Befehl `help`:

```
raptorxml help valany
```

Der obige Befehl enthält ein Argument: den Befehl `valany`, zu dem die Hilfe angezeigt werden soll. Bei Ausführung dieses Befehls werden die Hilfeinformationen zum Befehl `valany` angezeigt.

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Die Option `--help`
Die Hilfe zu einem Befehl kann auch durch die Option `--help` für diesen Befehl aufgerufen werden. Wenn Sie z.B. die Option `--help` folgendermaßen mit dem Befehl `valany` verwenden:

```
raptorxml valany --help
```

erzielen Sie dasselbe Ergebnis wie bei Verwendung des Befehls `help` mit einem Argument von `valany`:

```
raptorxml help valany
```

In beiden Fällen wird die Hilfe zum Befehl `valany` angezeigt.

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

 © 2014 Altova GmbH

3.5.2 Lizenzbefehle

Mit dem Befehl `licenseserver` wird RaptorXML Server auf Altova LicenseServer registriert. Als Argument erhält der Befehl den Namen oder die IP-Adresse des Servers, auf dem LicenseServer ausgeführt wird.

Windows **`RaptorXML licenseserver [options]`** ***`Server-Or-IP-Address`***
Linux **`raptorxml licenseserver [options]`** ***`Server-Or-IP-Address`***
Mac **`raptorxml licenseserver [options]`** ***`Server-Or-IP-Address`***

Nachdem RaptorXML Server erfolgreich auf LicenseServer registriert wurde, wird die URL der LicenseServer Web-Schnittstelle zurückgegeben. Geben Sie die URL in ein Browser-Fenster ein, um die Web-Schnittstelle aufzurufen und fahren Sie anschließend mit der Lizenzierung fort, wie in der LicenseServer-Dokumentation beschrieben.

Beispiel
Hier sehen Sie ein Beispiel für den Befehl `licenseserver`:

```
raptorxml licenseserver DOC.altova.com
```

Der Befehl gibt an, dass der Rechner namens `DOC.altova.com` der Rechner ist, auf dem Altova LicenseServer ausgeführt wird.

▼ Groß- und Kleinschreibung in der Befehlszeile

`RaptorXML` *unter Windows*
`raptorxml` *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Optionen
Die folgenden Optionen stehen zur Verfügung:

`--j|json=true|false`
Druckt das Ergebnis des Registrierungsversuchs als maschinenlesbares JSON-Objekt.

`--h|help`
Zeigt den Hilfetext zum Befehl an.

`--version`
Zeigt die Versionsnummer von RaptorXML Server an. Die Option sollte vor den Befehl gestellt werden. Beispiel: `raptorxml --version licenseserver`.

3.6 Lokalisierungsbefehle

Sie können für jede Sprache Ihrer Wahl eine lokalisierte Version der RaptorXML Applikation erstellen. Im Ordner `<ProgramFilesFolder>\Altova\RaptorXMLServer2015\bin\` stehen vier lokalisierte Versionen (Englisch, Deutsch, Spanisch und Japanisch) bereits zur Verfügung. Eine Lokalisierung für diese Sprache ist daher nicht mehr notwendig.

Um eine lokalisierte Version in einer anderen Sprache zu erstellen, gehen Sie folgendermaßen vor:

1. Generieren Sie mit Hilfe des Befehls `exportresourcestrings` eine XML-Datei, die die Ressourcenstrings enthält. Die Ressourcenstrings in dieser XML-Datei sind in einer der vier unterstützten Sprachen: je nachdem, welches Argument mit dem Befehl verwendet wird, in Englisch (`en`), Deutsch (`de`), Spanisch (`es`) oder Japanisch (`ja`).
2. Übersetzen Sie die Ressourcenstrings aus der Sprache der generierten XML-Datei in die Zielsprache. Die Ressourcenstrings bilden den Inhalt der `<string>` Elemente in der XML-Datei. Übersetzen Sie keine Variablen in geschweiften Klammern wie z.B. `{option}` oder `{product}`.
3. Wenden Sie sich an den Altova Support, um anhand Ihrer übersetzten XML-Datei eine lokalisierte RaptorXML DLL-Datei zu generieren.
4. Nachdem Sie Ihre lokalisierte DLL-Datei vom Altova Support erhalten haben, speichern Sie diese unter `<ProgramFilesFolder>\Altova\RaptorXMLServer2015\bin\`. Ihre DLL-Datei wird einen Namen in der Form `RaptorXMLServer_lc.dll` haben. Der `_lc` Teil des Namens enthält den Sprachencode. So steht z.B. in `RaptorXMLServer_de.dll` der `de` Teil für den Sprachencode für Deutsch.
5. Führen Sie den Befehl `setdeflang` aus, um Ihre lokalisierte DLL als die zu verwendende RaptorXML Applikation zu definieren. Verwenden Sie den Sprachencode, der Teil des DLL-Namens ist, als Argument des Befehls `setdeflang`.

Anmerkung: Altova RaptorXML Server ist mit Unterstützung für vier Sprachen erhältlich: Englisch, Deutsch, Spanisch und Japanisch. Sie müssen daher keine lokalisierte Version dieser Sprachen erstellen. Um eine dieser vier Sprachen als Standardsprache festzulegen, verwenden Sie den Befehl `setdeflang`.

© 2014 Altova GmbH

3.6.1 exportresourcestrings

Der Befehl **exportresourcestrings** gibt eine XML-Datei aus, die die RaptorXML Ressourcenstrings enthält. Der Befehl kann zwei Argumente haben: (i) die Sprache der Ressourcenstrings in der XML-Ausgabedatei und (ii) den Namen und Pfad der XML-Ausgabedatei. Derzeit unterstützte Exportsprachen sind (mit den Sprachcodes in Klammern): Englisch (en), Deutsch, (de), Spanisch (es) und Japanisch (ja).

Windows **RaptorXML exportresourcestrings *LanguageCode XMLOutputFile***
Linux **raptorxml exportresourcestrings *LanguageCode XMLOutputFile***
Mac **raptorxml exportresourcestrings *LanguageCode XMLOutputFile***

Argumente
Der Befehl exportresourcestrings kann die folgenden Argumente haben:

LanguageCode	Definiert die Sprache der Ressourcenstrings in der exportierten XML-Datei. Unterstützte Sprachen sind: en, de, es, ja
XMLOutputFile	Definiert den Namen und Pfad der exportierten XML-Datei.

Beispiel
Mit diesem Befehl wird unter c:\ eine Datei namens Strings.xml erstellt, die alle Ressourcenstrings der RaptorXML Applikation in deutscher Sprache enthält.

```
raptorxml exportresourcestrings de c:\Strings.xml
```

▼ Groß- und Kleinschreibung in der Befehlszeile

RaptorXML *unter Windows*
raptorxml *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (raptorxml) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (RaptorXML) nur unter Windows und Mac ausgeführt werden.

3.6.2 setdeflang

Der Befehl `setdeflang` (Kurzform ist `sdl`) definiert die Standardsprache von RaptorXML. Er akzeptiert ein obligatorisches Argument `LanguageCode`.

Windows **`RaptorXML setdeflang | sdl`** ***`LangaugeCode`***
Linux **`raptorxml setdeflang | sdl`** ***`LangaugeCode`***
Mac **`raptorxml setdeflang | sdl`** ***`LangaugeCode`***

Beispiel
Der obige Befehl definiert Deutsch als Standardsprache für Meldungen der Applikation.

```
raptorxml setdeflang de
```

▼ Groß- und Kleinschreibung in der Befehlszeile

`RaptorXML` *unter Windows*
`raptorxml` *unter Unix (Linux, Mac)*

* Beachten Sie, dass klein geschriebene Befehle (`raptorxml`) auf allen Plattformen (Windows, Linux und Mac) funktionieren, während großgeschriebene Befehle (`RaptorXML`) nur unter Windows und Mac ausgeführt werden.

Unterstützte Sprachen
In der nachstehenden Tabelle sind die derzeit unterstützten Sprachen zusammen mit ihren Sprachencodes aufgelistet.

`en`	Englisch
`de`	Deutsch
`es`	Spanisch
`ja`	Japanisch

 © 2014 Altova GmbH

3.7 Optionen

Dieser Abschnitt enthält eine Beschreibung aller CLI-Optionen, geordnet nach Funktionalität. Informationen darüber, welche Optionen mit den einzelnen Befehlen verwendet werden können, finden Sie in der Beschreibung zu den entsprechenden Befehlen.

- Kataloge, globale Ressourcen, ZIP-Dateien
- Meldungen, Fehler, Hilfe
- Verarbeitung
- XML
- XSD
- XQuery
- XSLT

3.7.1 Kataloge, globale Ressourcen, ZIP-Dateien

catalog

--catalog = *FILE*
Gibt den absoluten Pfad zu einer Root-Katalog-Datei an, die nicht die installierte Root-Katalog-Datei ist. Der Standardwert ist der absolute Pfad zur installierten Root-Katalog-Datei. (`<installationsordner>\Altova\RaptorXMLServer2015\etc\RootCatalog.xml`).
Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

user-catalog

--user-catalog = *FILE*
Definiert den absoluten Pfad zu einem XML-Katalog, der zusätzlich zum Root-Katalog verwendet werden soll. Informationen zum Arbeiten mit Katalogen finden Sie im Abschnitt XML-Kataloge.

enable-globalresources

--enable-globalresources = `true|false`
Aktiviert die globalen Ressourcen. Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

globalresourceconfig [gc]

--gc | --globalresourceconfig = *VALUE*
Definiert die aktive Konfiguration der globalen Ressource (und aktiviert globale Ressourcen).

globalresourcefile [gr]

--gr | --globalresourcefile = *FILE*
Definiert die globale Ressourcendatei (und aktiviert globale Ressourcen).

recurse

--recurse = `true|false`
Dient zur Auswahl von Dateien in einem ZIP-Archiv. Bei `true` wählt das Argument *InputFile* des Befehls die angegebene Datei auch in den Unterverzeichnissen aus. Beispiel: `test.zip|zip\test.xml` wählt Dateien mit dem Namen `test.xml` auf allen Ordnerebenen des ZIP-Ordners aus. Es können die Platzhalter `*` und `?` verwendet werden. Mit `*.xml` werden folglich alle `.xml` Dateien im ZIP-Ordner ausgewählt. Der Standardwert des Parameters ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

© 2014 Altova GmbH

3.7.2 Meldungen, Fehler, Hilfe, Timeout, Version

error-format

--error-format = `text|shortxml|longxml`
Definiert das Format der Fehlerausgabe. Der Standardwert ist `text`. Mit den anderen Optionen werden XML-Formate generiert, wobei mit `longxml` mehr Details generiert werden.

error-limit

--error-limit = *N*
Definiert das Fehlerlimit. Der Standardwert ist `100`. Zulässig sind Werte von `1` bis `999`. Dient dazu, die Prozessorverwendung während der Validierung einzuschränken. Bei Erreichung des Fehlerlimits wird die Validierung gestoppt.

help

--help
Zeigt den Hilfetext zum Befehl an. Beispiel: `valany --h`. (Alternativ dazu kann der Befehl `help` zusammen mit einem Argument verwendet werden. Beispiel: `help valany`.)

log-output

--log-output = *FILE*
Schreibt die Meldungsausgabe in die angegebene URL anstelle der Konsole. Stellen Sie sicher, dass das CLI Schreibrechte für den Ausgabepfad hat.

network-timeout

--network-timeout = *VALUE*
Definiert das Timeout in Sekunden für entfernte I/O-Operationen. Der Standardwert ist: `40`.

verbose

--verbose = `true|false`
Mit dem Wert `true` wird die Ausgabe zusätzlicher Informationen bei der Validierung aktiviert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

verbose-output

--verbose-output = *FILE*
Schreibt die ausführliche Ausgabe in *FILE*.

version

--version
Zeigt die Version von RaptorXML Server an. Setzen Sie `--version` bei Verwendung mit einem Befehl vor den Befehl.

3.7.3 Verarbeitung

listfile

--listfile = true|false
Bei true wird das Argument *InputFile* des Befehls als Textdatei behandelt, die einen Dateinamen pro Zeile enthält. Der Standardwert ist false. (Als Alternative können die Dateien im CLI getrennt durch ein Leerzeichen aufgelistet werden. Beachten Sie allerdings, dass CLIs eine maximale Zeichenanzahl haben.) Beachten Sie, dass die Option --listfile nur auf Argumente, nicht aber auf Optionen angewendet wird.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

parallel-assessment [pa]

--pa | --parallel-assessment = true|false
Bei Setzung auf true wird die Schema-Validierung parallel ausgeführt. Das bedeutet, wenn sich auf irgendeiner Ebene mehr als 128 Elemente befinden, so werden diese Elemente über mehrere Threads parallel verarbeitet. Auf diese Weise können besonders große XML-Dateien schneller verarbeitet werden, wenn diese Option aktiv ist. Parallele Validierungen können gleichzeitig auf einer hierarchischen Ebene ausgeführt werden, können in einem einzigen Infoset aber auch auf mehreren Ebenen erfolgen. Beachten Sie dass die parallele Validierung im Streaming-Modus nicht funktioniert. Aus diesem Grund wird die Option --streaming ignoriert, wenn --parallel-assessment auf true. gesetzt ist. Außerdem wird bei Verwendung der Option --parallel-assessment mehr Arbeitsspeicher benötigt. Die Standardeinstellung ist false. Die Kurzform für die Option ist --pa.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

script

--script = *FILE*
Führt nach Abschluss der Validierung das Python-Skript in der angegebenen Datei aus.

streaming

--streaming = true|false
Aktiviert die Streaming-Validierung. Standardwert ist true. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie --streaming auf den Wert false setzen). Wenn Sie die Option --script mit dem Befehl valxml-withxsd verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option --streaming ignoriert wird, wenn --parallel-assessment auf true gesetzt wird.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

 © 2014 Altova GmbH

3.7.4 XML

assessment-mode

--assessment-mode = `lax|strict`
Definiert den Beurteilungsmodus für die Gültigkeit von Schemas gemäß der XSD-Spezifikation. Der Standardwert ist `strict`. Das XML-Instanzdokument wird entsprechend dem mit dieser Option definierten Modus validiert.

dtd

--dtd = *FILE*
Definiert das für die Validierung zu verwendende externe DTD-Dokument. Wenn das XML-Dokument eine Referenz auf eine externe DTD enthält, setzt die CLI-Option die externe Referenz außer Kraft.

load-xml-with-psvi

--load-xml-with-psvi = `true|false`
Ermöglicht die Validierung von XML-Input-Dateien und die Generierung von Informationen für diese Dateien nach Validierung des Schemas. Der Standardwert ist: `false`.

namespaces

--namespaces = `true|false`
Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

xinclude

--xinclude = `true|false`
Aktiviert die Unterstützung für XML-Inkludierungen (XInclude). Der Standardwert ist `false`. Bei `false` werden die `include`-Elemente von XInclude ignoriert.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

xml-mode

--xml-mode = `wf|id|valid`
Definiert den zu verwendenden XML-Verarbeitungsmodus: `wf`=Wohlgeformtheitsprüfung; `id`=Wohlgeformtheitsprüfung mit ID/IDREF-Prüfung; `valid`=Validierung. Der Standardwert ist `wf`.

xsd

--xsd = *FILE*
Definiert ein oder mehrere XML-Schema-Dokumente, die für die Validierung von XML-Instanzdokumenten verwendet werden sollen. Um mehrere Schema-Dokumente zu definieren, fügen Sie die Option mehrmals hinzu.

3.7.5 XSD

assessment-mode

--assessment-mode = lax|strict
Definiert den Beurteilungsmodus für die Gültigkeit von Schemas gemäß der XSD-Spezifikation. Der Standardwert ist `strict`. Das XML-Instanzdokument wird entsprechend dem mit dieser Option definierten Modus validiert.

namespaces

--namespaces = true|false
Aktiviert die Verarbeitung unter Berücksichtigung des Namespace. Dies ist nützlich, um die XML-Instanz auf Fehler aufgrund falscher Namespaces zu überprüfen. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

schema-imports

--schema-imports = load-by-schemalocation | load-preferring-schemalocation | load-by-namespace | load-combining-both | license-namespace-only
Definiert das Verhalten von `xs:import` Elementen, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat: `<import namespace="someNS" schemaLocation="someURL">`. Sie verhalten sich folgendermaßen:

- `load-by-schemalocation`: Anhand des Werts des `schemaLocation` Attributs wird der Schemapfad ermittelt, wobei Katalog-Mappings berücksichtigt werden. Wenn das Namespace-Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `load-preferring-schemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalog-Mappings berücksichtigt werden. Falls kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalog-Mapping verwendet. Dies ist der **Standardwert**.
- `load-by-namespace`: Anhand des Werts des `namespace` Attributs wird der Schemapfad über ein Katalog-Mapping ermittelt.
- `load-combining-both`: Wenn entweder das Attribut `namespace` oder das Attribut `schemaLocation` ein Katalog-Mapping hat, so wird das Mapping verwendet. Wenn beide Attribute Katalog-Mappings haben, ist es vom Wert der Option `--schema-mapping` (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Falls kein Katalog-Mapping vorhanden ist, wird das `schemaLocation` Attribut verwendet.
- `license-namespace-only`: Der Namespace wird importiert. Kein Schema-Dokument wird importiert.

schema-location-hints

--schema-location-hints = load-by-schemalocation | load-by-namespace | load-combining-both | ignore

- Der Wert `load-by-schemalocation` verwendet die URL des Schemapfads in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML-Instanzdokumenten. Dies ist der **Standardwert**.
- Der Wert `load-by-namespace` verwendet den Namespace-Teil von `xsi:schemaLocation` und - im Fall von `xsi:noNamespaceSchemaLocation` - einen leeren String und ermittelt das Schema über ein Katalog-Mapping.
- Bei Verwendung von `load-combining-both` und wenn entweder der Namespace-Teil oder der URL-Teil ein Katalog-Mapping hat, so wird das Katalog-Mapping verwendet.

 © 2014 Altova GmbH

Wenn beide Katalog-Mappings haben, ist es vom Wert der `--schema-mapping` Option (XML/XSD-Option) abhängig, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalog-Mapping hat, wird die URL verwendet.

- Wenn der Wert der Option `ignore` ist, werden die beiden Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` ignoriert.

schema-mapping

`--schema-mapping =` `prefer-schemalocation | prefer-namespace`

Wenn eine der Optionen `--schemalocation-hints` oder `--schema-imports` einen Wert `load-combining-both` hat und wenn die betroffenen Namespace- und URL-Teile beide Katalog-Mappings haben, gibt der Wert dieser Option an, welches der beiden Mappings verwendet werden soll (das Namespace Mapping oder das URL-Mapping; der Wert `prefer-schemalocation` bezieht sich auf das URL-Mapping). Der Standardwert ist `prefer-schemalocation`.

xsd-version

`--xsd-version =` `1.0|1.1|detect`

Definiert die zu verwendende W3C Schema Definition Language (XSD) Version. Der Standardwert ist `1.0`. Diese Option eignet sich auch, um herauszufinden, inwiefern ein 1.0-kompatibles Schema nicht mit Schemaversion 1.1 kompatibel ist. Die Option `detect` ist eine Altova-spezifische Funktionalität. Mit dieser Option kann die Version des XML-Schema-Dokuments (`1.0` oder `1.1`) durch Lesen des Werts des `vc:minVersion` Attributs des `<xs:schema>` Elements des Dokuments ermittelt werden. Wenn der Wert des `@vc:minVersion` Attributs `1.1` ist, wird das Schema als Version `1.1` erkannt. Bei jedem anderen Wert wird das Schema als `1.0` erkannt. Bei jedem anderen Wert oder bei Fehlen des `@vc:minVersion` Attributs wird das Schema als Version `1.0` gelesen.

3.7.6 XQuery

indent-characters

--indent-characters = *VALUE*
Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

input

--input = *FILE*
Die URL der zu transformierenden XML-Datei.

keep-formatting

--keep-formatting = `true|false`
Behält die Formatierung des Zieldokuments so gut wie möglich bei. Der Standardwert ist: `true`.

omit-xml-declaration

--omit-xml-declaration = `true|false`
Serialisierungsoption, mit der angegeben wird, ob die XML-Deklaration in der Ausgabe enthalten sein soll oder nicht. Wenn der Wert `true` ist, enthält das Ausgabedokument keine XML-Deklaration. Wenn der Wert `false` ist, wird eine XML-Deklaration inkludiert. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

output

output = *FILE*
Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Wenn keine `--output` Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

output-encoding

--output-encoding = *VALUE*
Der Wert des Kodierungsattributs im Ausgabedokument. Gültige Werte sind die Namen im IANA-Zeichensatz-Register. Der Standardwert ist `UTF-8`.

output-indent

--output-indent = `true|false`
Wenn der Wert `true` ist, wird die Ausgabe entsprechend ihrer hierarchischen Struktur eingerückt. Bei `false` gibt es keine hierarchische Einrückung. Der Standardwert ist `false`.
Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

output-method

--output-method = `xml|html|xhtml|text`
Definiert das Ausgabeformat. Der Standardwert ist `xml`.

param [p]

 © 2014 Altova GmbH

--p | --param = *KEY:VALUE*

XQuery

Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der `declare variable` Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort `external`, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts `external` wird `$foo` zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

XSLT

Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

updated-xml

--updated-xml = `discard|writeback|asmainresult`

Definiert, wie die aktualisierte XML-Datei behandelt werden soll. Es gibt folgende Möglichkeiten:

- Die Aktualisierungen können verworfen und nicht in die Datei geschrieben werden (`discard`).
- die Aktualisierungen können zurück in die mit der Option `--input` definierte XML-Input-Datei geschrieben werden (`writeback`).
- Die Aktualisierungen können entweder im Standardausgabedokument oder unter dem mit der Option `--output` definierten Pfad gespeichert werden (falls dieser definiert ist).

Der Standardwert ist: `discard`.

xquery-update-version

--xquery-update-version = `1|3`

Definiert, ob der XQuery-Prozessor XQuery Update Facility 1.0 oder XQuery Update Facility 3.0 verwenden soll. Der Standardwert ist `3`.

xquery-version

--xquery-version = `1|3`

Gibt an, ob der XQuery-Prozessor XQuery 1.0 oder XQuery 3.0 verwenden soll. Der Standardwert ist `1`.

 © 2014 Altova GmbH

3.7.7 XSLT

chartext-disable

--chartext-disable = true|false
Deaktiviert Diagrammerweiterungen. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

dotnetext-disable

--dotnetext-disable = true|false
Deaktiviert .NET-Erweiterungen. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

indent-characters

--indent-characters = *VALUE*
Definiert den Zeichenstring, der als Einrückung verwendet werden soll.

input

--input = *FILE*
Die URL der zu transformierenden XML-Datei.

javaext-barcode-location

--javaext-barcode-location = *FILE*
Definiert den Pfad der Barcode-Erweiterungsdatei.

javaext-disable

--javaext-disable = true|false
Deaktiviert Java-Erweiterungen. Der Standardwert ist false.
Hinweis: Die Booleschen Optionswerte werden auf true gesetzt, wenn die Option ohne einen Wert definiert wird.

output

output = *FILE*
Die URL der primären Ausgabedatei. So ist z.B. im Fall der Ausgabe mehrerer HTML-Dateien die primäre Ausgabedatei der Pfad der Eintrittspunkt-HTML-Datei. Wenn keine --output Option definiert ist, wird die Ausgabe in die Standardausgabe geschrieben.

param [p]

--p | --param = *KEY:VALUE*

XQuery

Definiert den Wert eines externen Parameters. Ein externer Parameter ist im XQuery-Dokument mit der declare variable Deklaration gefolgt von einem Variablennamen und anschließend dem Schlüsselwort external, gefolgt von einem Semikolon deklariert. Beispiel:

```
declare variable $foo as xs:string external;
```

Aufgrund des Schlüsselworts external wird $foo zu einem externen Parameter, dessen Wert zur Laufzeit von einer externen Quelle aus übergeben wird. Der externe

Parameter erhält mit dem CLI-Befehl einen Wert. Beispiel:

```
--param=foo:'MyName'
```

In der obigen Beschreibungsanweisung ist *KEY* der Name des externen Parameters, *VALUE* der als XPath-Ausdruck angegebene Wert des externen Parameters. Im CLI verwendete Parameter müssen im XQuery-Dokument deklariert sein. Wenn mehrere externe Parameter als Werte an das CLI übergeben werden, muss jeder eine separate `--param` Option erhalten. Wenn der XPath-Ausdruck Leerzeichen enthält, muss er in doppelte Anführungszeichen gesetzt werden.

XSLT

Definiert einen globalen Stylesheet-Parameter. *KEY* ist der Parametername, *VALUE* der als XPath-Ausdruck angegebene Parameterwert. Im CLI verwendete Parameter müssen im Stylesheet deklariert sein. Wenn mehrere Parameter verwendet werden, muss vor jedem Parameter die `--param` Option verwendet werden. Wenn der XPath-Ausdruck Leerzeichen enthält - ob im XPath-Ausdruck selbst oder in einem String-Literal im Ausdruck - muss er in doppelte Anführungszeichen gesetzt werden. Beispiel:

```
raptorxml xslt --input=c:\Test.xml --output=c:\Output.xml --
param=date://node[1]/@att1 --p=title:'stringwithoutspace' --
param=title:"'string with spaces'" --p=amount:456 c:\Test.xslt
```

streaming

--streaming = `true|false`

Aktiviert die Streaming-Validierung. Standardwert ist `true`. Die im Arbeitsspeicher gehaltene Datenmenge wird im Streaming-Modus minimiert. Der Nachteil ist, dass später eventuell benötigte Informationen - z.B. ein Datenmodell des XML-Instanzdokuments - nicht mehr verfügbar sind. In Situationen, in denen dies eine Rolle spielt, muss der Streaming-Modus deaktiviert werden (indem Sie `--streaming` auf den Wert `false` setzen). Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, sollten Sie das Streaming deaktivieren. Beachten Sie, dass die Option `--streaming` ignoriert wird, wenn `--parallel-assessment` auf `true` gesetzt wird.

Hinweis: Die Booleschen Optionswerte werden auf `true` gesetzt, wenn die Option ohne einen Wert definiert wird.

template-entry-point

--template-entry-point = *VALUE*

Gibt den Namen einer benannten Vorlage im XSLT-Stylesheet an, das der Eintrittspunkt der Transformation ist.

template-mode

--template-mode = *VALUE*

Definiert den Vorlagenmodus für die Transformation.

xslt-version

--xslt-version = `1|2|3`

Definiert, ob der XSLT-Prozessor XSLT 1.0, XSLT 2.0 oder XSLT 3.0 verwenden soll. Der Standardwert ist `3`.

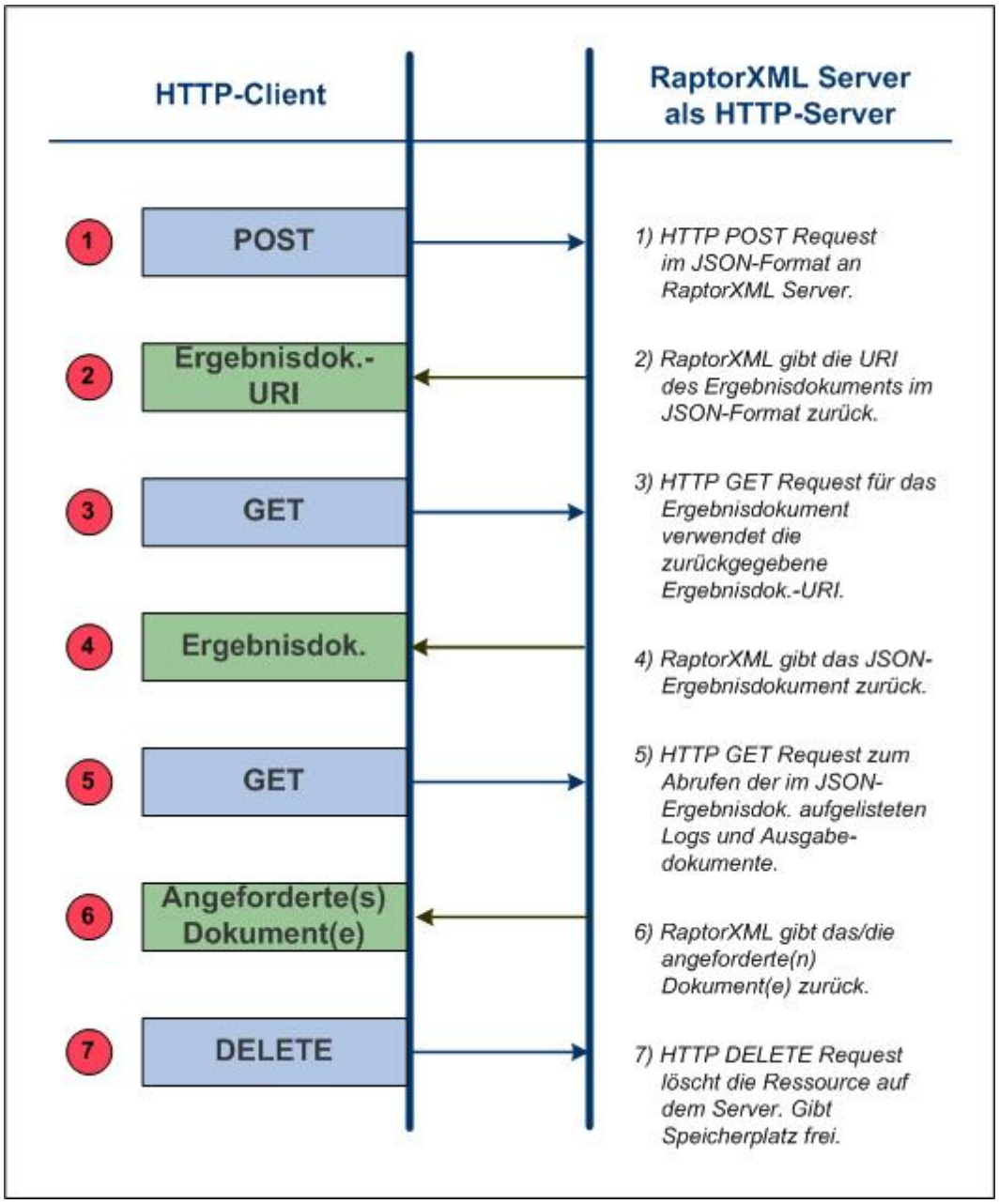
 © 2014 Altova GmbH

Kapitel 4

HTTP-Schnittstelle

4 HTTP-Schnittstelle

RaptorXML Server nimmt über HTTP gesendete Validierungsaufträge an. Die Übertragung der Auftragsbeschreibung sowie der Ergebnis erfolgt im JSON-Format. Im unten gezeigten Diagramm sehen Sie eine Beschreibung des Ablaufs.



 © 2014 Altova GmbH

Sicherheitsfragen im Zusammenhang mit der HTTP-Schnittstelle
Standardmäßig können Ergebnisdokumente über die HTTP-Schnittstelle in jeden durch den Client angegebenen Ordner (auf den über das HTTP-Protokoll Zugriff besteht) geschrieben werden. Beim Konfigurieren von RaptorXML Server sollte dieser Sicherheitsaspekt daher berücksichtigt werden.

Falls die Sicherheit eventuell gefährdet sein könnte oder jemand über die Schnittstelle unbefugten Zugriff erhalten könnte, kann der Server so konfiguriert werden, dass Ergebnisdokumente in ein eigenes Ausgabeverzeichnis auf dem Server selbst geschrieben werden. Dies geschieht durch Setzen der Option `server.unrestricted-filesystem-access` der Server-Konfigurationsdatei auf `false`. Wenn der Zugriff auf diese Weise eingeschränkt ist, kann der Client Ergebnisdokumente aus dem dafür vorgesehenen Ausgabeverzeichnis mit `GET` Requests herunterladen. Alternativ dazu kann ein Administrator die Ergebnisdokumentdateien vom Server in den Zielordner kopieren/laden.

In diesem Abschnitt
Bevor Sie einen Client Request senden, muss RaptorXML Server gestartet und ordnungsgemäß konfiguriert werden. Eine Beschreibung dazu finden Sie im Abschnitt Einrichten des Servers. Eine Beschreibung dazu, wie man Client Requests sendet, finden Sie im Abschnitt Client Requests.

4.1 Einrichten des Servers

Um RaptorXML Server einzurichten, gehen Sie vor, wie im Folgenden beschrieben. Es wird davon ausgegangen, dass RaptorXML Server bereits korrekt installiert und lizenziert wurde.

1. Damit RaptorXML Server über HTTP aufgerufen werden kann, muss das Produkt entweder als Dienst oder als Applikation gestartet sein. Die Vorgehensweisen unterscheiden sich je nach Betriebssystem und sind hier für die folgenden Betriebssysteme beschrieben: Windows, Linux, Mac OS.
2. Verwenden Sie die Server-Anfangskonfiguration, um die Verbindung zum Server zu testen. (Die Server-Anfangskonfiguration ist die bei der Installation vordefinierte Standardkonfiguration.) Mit einem einfachen HTTP `GET` Request wie z.B. `http://localhost:8087/v1/version` können Sie die Verbindung testen. (Sie können den Request auch in die Adressleiste Ihres Browser-Fensters eingeben.) Wenn der Dienst ausgeführt wird, sollten Sie eine Antwort auf den HTTP Test-Request, wie z.B. den Version-Request oben, erhalten.
3. Öffnen Sie die Server-Konfigurationsdatei, `server_config.xml`. Wenn Sie Einstellungen in der Datei ändern möchten, bearbeiten Sie die Server-Konfigurationsdatei und speichern Sie die Änderungen.
4. Falls Sie die Server-Konfigurationsdatei bearbeitet haben, starten Sie den RaptorXML Server als Dienst neu, damit die neuen Konfigurationseinstellungen angewendet werden. Testen Sie anschließend die Verbindung erneut, um sicherzustellen, dass der Dienst ausgeführt wird und Sie ihn aufrufen können.

Anmerkung: Fehler, die beim Start des Servers auftreten, die verwendete Server-Konfigurationsdatei sowie Lizenzierungsfehler werden im System-Log protokolliert. Wenn Probleme mit dem Server auftreten, konsultieren Sie bitte das System-Log.

 © 2014 Altova GmbH

4.1.1 Starten des Servers

In diesem Abschnitt werden folgende Schritte beschrieben:

- Pfad zur ausführbaren Server-Datei
- Starten von RaptorXML als Dienst unter Windows
- Starten von RaptorXML als Dienst unter Linux
- Starten von RaptorXML als Dienst unter Mac OS

Pfad zur ausführbaren Server-Datei
Die ausführbare RaptorXML Server-Datei ist standardmäßig im folgenden Ordner installiert:

```
<ProgramFilesFolder>\Altova\RaptorXMLServer2015\bin\RaptorXML.exe
```

Über die ausführbare Datei kann RaptorXML Server als Dienst gestartet werden.

Starten als Dienst unter Windows
Bei der Installation wurde RaptorXML Server als Dienst unter Windows registriert. Sie müssen RaptorXML Server allerdings als **Dienst starten**. Dazu gibt es die folgenden Möglichkeiten:

- über den Altova ServiceController, der in der Task-Leiste als Symbol zur Verfügung steht. Falls das Symbol nicht angezeigt wird, können Sie Altova ServiceController starten und sein Symbol zur Task-Leiste hinzufügen. Gehen Sie dazu zum **Startmenü** und wählen Sie **Alle Programme | Altova | Altova LicenseServer | Altova ServiceController**.
- über die Verwaltungskonsole für Windows-Dienste: **Systemsteuerung | Alle Systemsteuerungselemente | Verwaltung | Dienste**.
- über das Eingabeaufforderungsfenster, wenn es mit Administratorrechten geöffnet wurde. Verwenden Sie von jedem beliebigen Verzeichnis aus den folgenden Befehl: **`net start "Altova RaptorXML Server"`**
- über die ausführbare RaptorXML Server-Datei in einem Eingabeaufforderungsfenster: **`RaptorXMLServer.exe debug`**. Daraufhin wird der Server gestartet, wobei Informationen über die Serveraktivitäten direkt im Eingabeaufforderungsfenster angezeigt werden. Sie können die Informationen über die Serveraktivität über die Einstellung `http.log-screen` der Server-Konfigurationsdatei ein- und ausblenden. Drücken Sie **Strg+Untbr** (oder **Strg +Pause**), um den Server zu beenden. Wenn der Server anstatt als Dienst, wie im vorigen Schritt beschrieben, auf diese Art gestartet wird, wird er beendet, wenn das Eingabeaufforderungsfenster geschlossen wird oder sich der Benutzer abmeldet.

Starten als Dienst unter Linux
Starten Sie RaptorXML Server mit dem folgenden Befehl als Dienst:

```
[Debian]   sudo /etc/init.d/raptorxmlserver start
```

```
[Ubuntu]   sudo initctl start raptorxmlserver
[CentOS]   sudo initctl start raptorxmlserver
[RedHat]   sudo initctl start raptorxmlserver
```

Um RaptorXML Server zu beenden, verwenden Sie:

```
[Debian]   sudo /etc/init.d/raptorxmlserver stop
[Ubuntu]   sudo initctl stop raptorxmlserver
[CentOS]   sudo initctl stop raptorxmlserver
[RedHat]   sudo initctl stop raptorxmlserver
```

Starten als Dienst unter Mac OS X

Starten Sie RaptorXML Server mit dem folgenden Befehl als Dienst:

```
sudo launchctl load /Library/LaunchDaemons/
com.altova.RaptorXMLServer2015.plist
```

Um RaptorXML Server zu beenden, verwenden Sie:

```
sudo launchctl unload /Library/LaunchDaemons/
com.altova.RaptorXMLServer2015.plist
```

 © 2014 Altova GmbH

4.1.2 Testen der Verbindung

In diesem Abschnitt werden folgende Schritte beschrieben:

- GET Request zum Testen der Verbindung
- Server Response und Empfangsbereitschaft für JSON-Datenstruktur

---

GET Request zum Testen der Verbindung
Nachdem RaptorXML Server gestartet wurde, testen Sie die Verbindung mit einem `GET` Request. (Sie können diesen Request auch in die Adressleiste eines Browser-Fensters eingeben.)

```
http://localhost:8087/v1/version
```

Anmerkung: Die Schnittstelle und die Port-Nummer von RaptorXML Server ist in der Server-Konfigurationsdatei, `server_config.xml`, definiert. Eine Beschreibung dazu finden Sie im nächsten Abschnitt Server-Konfiguration.

Server Response und Empfangsbereitschaft für JSON-Datenstruktur
Wenn der Dienst ausgeführt wird und der Server richtig konfiguriert ist, sollte der Request immer funktionieren. RaptorXML Server gibt seine Versionsinformationen als JSON-Datenstruktur zurück (*Codefragment unten*).

```
{
  "copyright": "Copyright (c) 1998-2013 Altova GmbH. ...",
  "name": "Altova RaptorXML+XBRL Server 2013 rel. 2 sp1",
  "eula": "http://www.altova.com/server_software_license_agreement.html"
}
```

Anmerkung: Wenn Sie die Server-Konfiguration durch Bearbeiten der Server-Konfigurationsdatei ändern, sollten Sie die Verbindung erneut testen.

4.1.3 Konfigurieren des Servers

In diesem Abschnitt werden folgende Schritte beschrieben:

- Server-Konfigurationsdatei: Anfangseinstellungen
- Server-Konfigurationsdatei: Ändern der Anfangseinstellungen, Zurücksetzen auf die Anfangseinstellungen
- Server-Konfigurationsdatei: Empfangsbereitschaft und Einstellungen
- Server-Konfigurationsdatei: Beschreibung der Einstellungen
- Konfigurieren der Server-Adresse

Server-Konfigurationsdatei: Anfangseinstellungen
RaptorXML Server wird mit Hilfe einer Konfigurationsdatei namens `server_config.xml` konfiguriert. Diese Datei befindet sich standardmäßig unter:

```
C:\Program Files (x86)\Altova\RaptorXMLServer2015\etc\server_config.xml
```

In der Anfangskonfiguration für RaptorXML Server sind die folgenden Einstellungen definiert:

- die Port-Nummer `8087` als Port für den Server.
- dass der Server nur lokale Verbindungen (`localhost`) empfängt.
- dass der Server die Ausgabedatei in folgenden Ordner schreibt: `C:\ProgramData\Altova\RaptorXMLServer2015\Output\`.

Andere Standardeinstellungen sind unter Empfangsbereitschaft von `server_config.xml` weiter unten beschrieben.

Server-Konfigurationsdatei: Ändern der Anfangseinstellungen, Zurücksetzen auf die Anfangseinstellungen
Wenn Sie die Anfangseinstellungen ändern möchten, müssen Sie die Server-Konfigurationsdatei, `server_config.xml` (siehe Empfangsbereitschaft unten) ändern, diese speichern und anschließend RaptorXML Server als Dienst neu starten.

Wenn Sie die ursprüngliche Server-Konfigurationsdatei wiederherstellen möchten, sodass wieder die Anfangseinstellungen für den Server konfiguriert sind), führen Sie den Befehl `createconfig` aus:

```
RaptorXML.exe createconfig
```

Bei Ausführung dieses Befehls werden die Anfangseinstellungen wiederhergestellt und die Datei `server_config.xml` wird damit überschrieben. Der Befehl `createconfig :-)` ist nützlich, wenn Sie die Server-Konfiguration auf die Anfangseinstellungen zurücksetzen möchten.

 © 2014 Altova GmbH

Server-Konfigurationsdatei: Empfangsbereitschaft und Einstellungen
Unten sehen Sie den Inhalt der Server-Konfigurationsdatei, **server_config.xml**, mit den Anfangseinstellungen. Die Einstellungen in dieser Datei sind unterhalb des Codes erläutert.

server_config.xml

```
<config xmlns="http://www.altova.com/schemas/altova/raptorxml/config"
  xsi:schemaLocation="http://www.altova.com/schemas/altova/raptorxml/config
  http://www.altova.com/schemas/altova/raptorxml/config.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <language>en</language>
  <server.unrestricted-filesystem-access>true</server.unrestricted-filesystem-
  access>
  <server.output-root-dir>C:\ProgramData\Altova\RaptorXMLServer2015\Output\</
  server.output-root-dir>
  <server.script-root-dir>C:\Program Files (x86)\Altova\RaptorXMLServer2015\etc
  \scripts\</server.script-root-dir>
  <!--<server.catalog-file>catalog.xml</server.catalog-file>-->
  <server.log-file>C:\ProgramData\Altova\RaptorXMLServer2015\Log\server.log</
  server.log-file>
  <http.environment>production</http.environment>
  <!--<http.socket-host>localhost</http.socket-host>-->
  <http.socket-port>8087</http.socket-port>
  <http.log-screen>true</http.log-screen>
  <http.access-file>C:\ProgramData\Altova\RaptorXMLServer2015\Log\access.log</
  http.access-file>
  <http.error-file>C:\ProgramData\Altova\RaptorXMLServer2015\Log\error.log</
  http.error-file>

</config>
```

Einstellungen

language
Definiert die Sprache von Server-Meldungen in einem optionalen `language` Element. Der Standardwert ist `en` (Englisch). Zulässige Werte sind `en|de|es|ja` (Englisch, Deutsch, Spanisch, Japanisch). Eine kurze Einführung, wie Sie RaptorXML lokalisieren, finden Sie unter Lokalisierungsbefehle.

server.unrestricted-filesystem-access
Wenn die Option auf `true` (den Standardwert) gesetzt ist, werden die Ausgabedateien direkt in den vom Benutzer und in Python-Skripts angegebenen Ordner geschrieben (wobei eventuell Dateien desselben Namens überschrieben werden). Wenn der Wert auf `false` gesetzt wird, werden die Dateien im Ausgabeverzeichnis in das Verzeichnis für den Auftrag geschrieben und die URI der Datei wird in das Ergebnisdokument inkludiert. Wenn Sie den Wert auf `false` setzen, ist die Sicherheitsstufe etwas höher, da die Dateien nur in ein bestimmtes Verzeichnis auf dem Server geschrieben werden können. Die Auftragsausgabedateien können anschließend überprüft

und in einen anderen Ordner kopiert werden.

server.output-root-dir
Das Verzeichnis, in dem die Ausgabedateien aller gesendeten Aufträge gespeichert werden.

server.script-root-dir
Das Verzeichnis, in dem vertrauenswürdige Python Skripts gespeichert werden sollen. Die Option `script` funktioniert bei Verwendung über die HTTP-Schnittstelle nur dann, wenn Skripts aus diesem vertrauenswürdigen Verzeichnis verwendet werden. Wird ein Python Skript aus einem anderen Verzeichnis angegeben, wird ein Fehler zurückgegeben. *Siehe 'Python-Skripts sicher machen'*.

server.catalog-file
Die URL der zu verwendenden XML-Katalogdatei. Standardmäßig wird die Katalogdatei `RootCatalog.xml` aus dem Ordner `<ProgramFilesFolder>\Altova\RaptorXMLServer2015\etc` verwendet. Verwenden Sie die `server.catalog-file` Einstellung nur dann, wenn Sie die Standard-Katalogdatei ändern möchten.

server.log-file
Der Name und Pfad der Server Log-Datei. Die Ereignisse auf dem Server wie z.B. *Server gestartet/beendet* werden ständig im Ereignis-Log des Systems protokolliert und in einem Ansichtsprogramm für Systemereignisse wie z.B. Windows Event Viewer angezeigt. Zusätzlich zur Anzeige im Ansichtsprogramm können Log-Meldungen auch in die mit der Option `server.log-file` definierte Datei geschrieben werden. Die Server-Log-Datei enthält Informationen über alle Aktivitäten auf dem Server wie z.B. Fehler beim Server-Start, die verwendeten Konfigurationsdateien und Lizenzfehler.

http.environment
Die internen Umgebungen von raptorxml: `production| development`. Die Development-Umgebung ist mehr auf die Bedürfnisse von Entwicklern ausgerichtet und ermöglicht einfacheres Debuggen als über die Production-Umgebung.

http.socket-host
Die Schnittstelle, über die RaptorXML Server aufgerufen wird. Falls RaptorXML Server auch Verbindungen mit entfernten Rechnern gestatten soll, kommentieren Sie das Element ein und setzen Sie den Inhalt auf: `0.0.0.0`. Das Element sollte dann folgendermaßen lauten: `<http.socket-host>0.0.0.0</http.socket-host>`. Auf diese Weise kann der Dienst auf jeder adressierbaren Schnittstelle des Server-Rechners gehostet werden. Stellen Sie in diesem Fall sicher, dass die Firewall-Einstellungen entsprechend konfiguriert sind. Eingehende Firewall Ausnahmen für Altova-Produkte müssen folgendermaßen registriert werden: Altova LicenseServer: Port `8088`; Altova RaptorXML Server: Port `8087`; Altova FlowForce Server: Port `8082`.

http.socket-port
Der Port, über den der Dienst aufgerufen wird. Der Port muss festgelegt und bekannt sein, damit HTTP Requests korrekt an den Dienst adressiert werden können.

 © 2014 Altova GmbH

http.log-screen
Wenn RaptorXML Server mit dem Befehl `RaptorXMLServer.exe debug`, (siehe Starten des Servers) gestartet wird, und wenn `http.log-screen` auf `true` gesetzt ist, so wird die Serveraktivität im Eingabeaufforderungsfenster angezeigt. Andernfalls wird die Serveraktivität nicht angezeigt. Zusätzlich zum Schreiben von Log-Dateien, wird der Log-Bildschirm angezeigt.

http.access-file
Der Name und Pfad der HTTP-Zugriffsdatei. Die Zugriffsdatei enthält Informationen über Aktivitäten im Zusammenhang mit dem Zugriff. Sie enthält Informationen, die beim Beheben von Verbindungsproblemen hilfreich sein können.

http.error-file
Der Name und Pfad der HTTP-Fehlerdatei. Die Fehlerdatei enthält Fehler im Zusammenhang mit dem Netzwerkverkehr von und zum Server. Bei Verbindungsproblemen kann diese Datei nützliche Informationen für deren Behebung enthalten.

Die RaptorXML Server Server-Adresse
Die HTTP-Adresse des Servers besteht aus dem Sockt-Host und dem Socket-Port:

```
http://{socket-host}:{socket-port}/
```

In der Anfangskonfiguration lautet die Adresse folgendermaßen:

```
http://localhost:8087/
```

Um die Adresse zu ändern, ändern Sie in der Server-Konfigurationsdatei `server_config.xml` die Einstellungen von `http.socket-host` und `http.socket-port`. Angenommen, der Server-Rechner hat die IP-Adresse `100.60.300.6` und es wurden die folgenden Sever-Konfigurationseinstellungen vorgenommen:

```
<http.socket-host>0.0.0.0</http.socket-host>
<http.socket-port>8087</http.socket-port>
```

RaptorXML Server kann in diesem Fall folgendermaßen adressiert werden:

```
http://100.60.300.6:8087/
```

Anmerkung: Nach Änderung von `server_config.xml` muss RaptorXML Server neu gestartet werden, damit die neuen Werte angewendet werden.

Anmerkung: Bei Problemen mit der Verbindung zu RaptorXML Server können die Informationen in den in `http.access-file` und `http.error-file` definierten Dateien beim Beheben der Probleme helfen.

Anmerkung: Nachrichten, die an RaptorXML Server gesendet werden, müssen Pfadnamen enthalten, die auf dem Server-Rechner gültig sind. Dokumente auf dem Server-Rechner

können entweder lokal oder entfernt (z.B. über HTTP URIs) aufgerufen werden.

© 2014 Altova GmbH

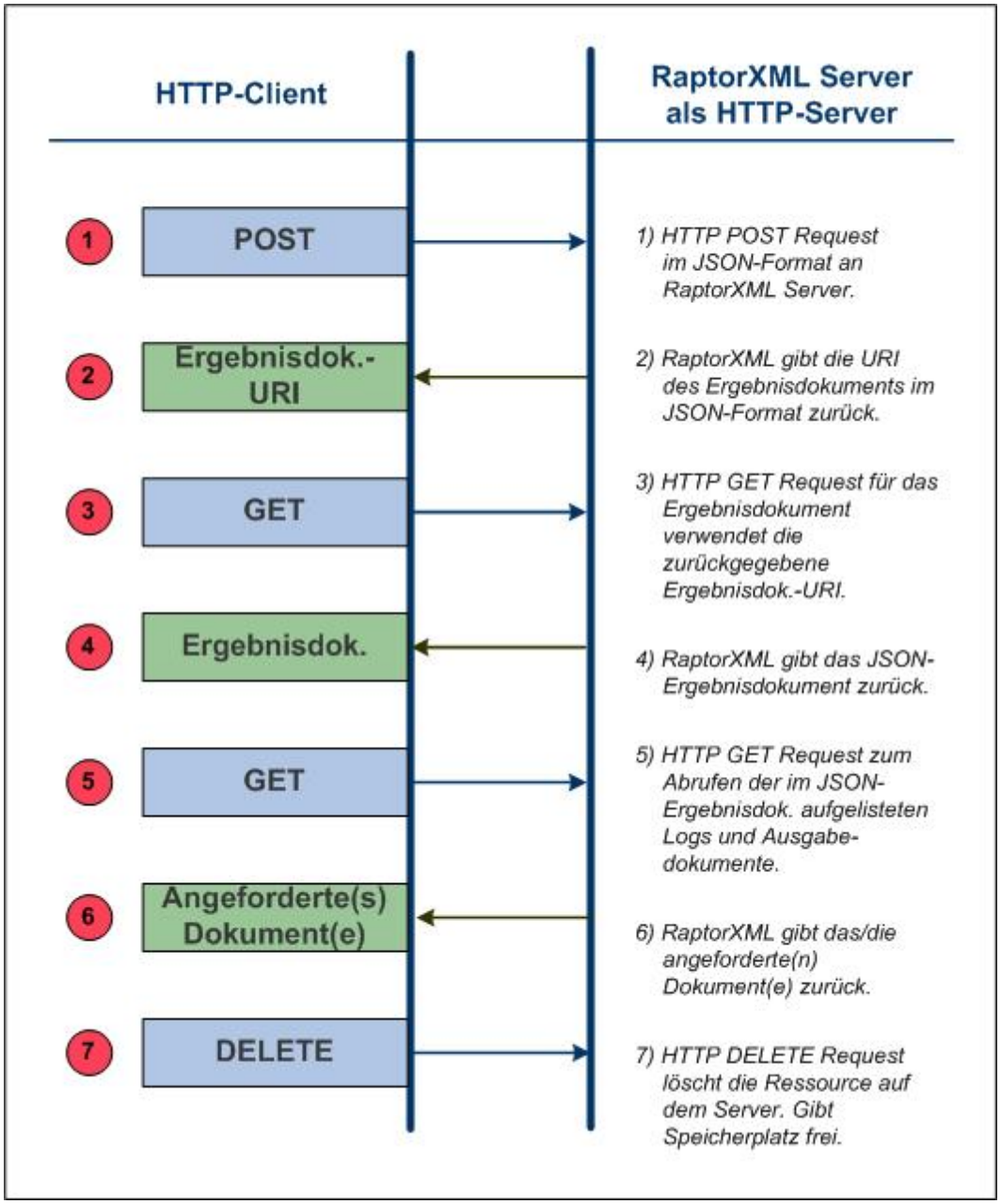
4.2 Client Requests

Nachdem RaptorXML Server als Dienst gestartet wurde, stehen seine Funktionalitäten jedem HTTP-Client zur Verfügung. Der HTTP-Client kann:

- die HTTP-Methoden `GET`, `PUT`, `POST` und `DELETE` verwenden
- das `Content-Type` Header-Feld definieren

Benutzerfreundlicher HTTP-Client
Im Internet steht ein ganze Reihe von Web Clients zum Download zur Verfügung. Wir haben RESTClient von Mozilla, einen benutzerfreundlichen und zuverlässigen Web Client, verwendet. Dieser Client kann als Plug-in zu Firefox hinzugefügt werden, ist einfach zu installieren, unterstützt die von RaptorXML benötigen HTTP-Methoden und bietet eine ausreichend gute JSON-Syntaxfärbung. Wenn Sie bisher noch nicht mit HTTP-Clients gearbeitet haben, empfehlen wir Ihnen RESTClient. Beachten Sie bitte, dass Sie RESTClient auf eigenes Risiko installieren und verwenden.

Ein typischer Client Request besteht, wie im Diagramm unten gezeigt, aus einer Reihe von Schritten.



Im Folgenden finden Sie einige wichtige Anmerkungen zu den einzelnen Schritten. Schlüsselbegriffe sind fett gedruckt.

1. Mit Hilfe einer HTTP `POST`-Methode wird ein Request im **JSON-Format** erstellt. Der Request könnte für jede Funktionalität von RaptorXML Server sein. So könnte z.B. eine Validierung oder eine XSLT-Transformation angefordert werden. Die im Request verwendeten Befehle, Argumente und Optionen sind dieselben, die auch in der Befehlszeile verwendet werden. Der Request wird auf: **`http://localhost:8087/v1/queue`** mittels `POST` bereitgestellt, wobei **`localhost:8087`** hier die Adresse von RaptorXML Server (die in der Anfangskonfiguration verwendete Server-Adresse) ist. Ein solcher Request wird als **RaptorXML Server-Auftrag** bezeichnet.

 © 2014 Altova GmbH

2. Wenn der Request von RaptorXML Server erhalten und für die Verarbeitung akzeptiert wurde, wird nach Verarbeitung des Auftrags ein **Ergebnisdokument** mit den Ergebnissen der Server-Aktion erstellt. Die **URI dieses Ergebnisdokuments** (im Diagramm oben die Ergebnisdok.-URI) wird an den Client zurückgegeben. Beachten Sie, dass die URI unmittelbar nach Übernahme des Auftrags für die Verarbeitung (nachdem er in die Warteschlange gestellt wurde) und auch, wenn die Verarbeitung noch nicht abgeschlossen wurde, zurückgegeben wird.

3. Der Client sendet (über die Ergebnisdokument-URI) in einer `GET`-Methode einen Request für das Ergebnisdokument an den Server. Wenn der Auftrag zum Zeitpunkt des Empfangs des Request noch nicht gestartet oder noch nicht abgeschlossen wurde, gibt der Server den Status *Running* zurück. Der `GET` Request muss so oft wiederholt werden, bis der Auftrag fertig gestellt ist und das Ergebnisdokument erstellt wurde.

4. RaptorXML Server gibt das Ergebnisdokument im JSON-Format zurück. Das Ergebnisdokument kann die **URIs von Fehler- oder Ausgabedokumenten,** die von RaptorXML Server beim Verarbeiten des ursprünglichen Request erzeugt wurden, enthalten. So werden z.B. Fehlerprotokolle zurückgegeben, wenn bei einer Validierung Fehler ausgegeben wurden. Die primären Ausgabedokumente, wie z.B. das Ergebnis einer XSLT-Transformation werden zurückgegeben, wenn der Auftrag zur Erzeugung einer Ausgabe erfolgreich ausgeführt wurde.

5. Der Client sendet die URIs der in Schritt 4 erhaltenen Ausgabedokumente über eine HTTP `GET`-Methode an den Server. Jeder Request wird in einer separaten `GET`-Methode gesendet.

6. RaptorXML Server gibt die angeforderten Dokumente in Antwort auf die in Schritt 5 gesendeten `GET`-Requests zurück.

7. Der Client kann nicht benötigte Dokumente, die als Ergebnis eines Auftrags-Request auf dem Server generiert wurden, löschen. Zu diesem Zweck sendet er die URI des entsprechenden Ergebnisdokuments in einer HTTP `DELETE`-Methode. Daraufhin werden alle im Zusammenhang mit diesem Auftrag generierten Dateien von der Festplatte gelöscht. Dazu gehören das Ergebnisdokument, alle temporären Dateien sowie alle Fehler- und Ausgabedokumentdateien. Dadurch schaffen Sie Platz auf der Festplatte des Servers.

In den Unterabschnitten dieses Abschnitts werden die einzelnen Schritte näher beschrieben.

4.2.1 Initiieren von Aufträgen mittels POST

In diesem Abschnitt werden folgende Schritte beschrieben:

- Senden des Request
- JSON-Syntax für POST Requests
- Hochladen von Dateien mit dem POST Request

Senden des Request
Ein RaptorXML Server-Auftrag wird mit der HTTP `POST` -Methode initiiert.

HTTP-Methode	URI	Content-Type	Body
`POST`	`http://localhost:8087/v1/queue/`	`application/json`	JSON

Beachten Sie die folgenden Punkte:

- Die obige URI hat eine Server-Adresse, für die die Einstellungen der Anfangskonfiguration verwendet werden.
- Die URI hat einen `/v1/queue/` Pfad, der in der URI vorhanden sein muss. Dabei handelt es sich um einen abstrakten Ordner im Arbeitsspeicher, in den der Auftrag platziert wird.
- Die richtige Versionsnummer `/vN` ist diejenige, die der Server zurückgibt (und nicht notwendigerweise die in der Dokumentation verwendete). Die Nummer, die der Server zurückgibt, ist die Versionsnummer der aktuellen HTTP-Schnittstelle. Frühere Versionsnummern stehen für ältere Versionen der HTTP-Schnittstelle, die weiterhin aus Gründen der Rückwärtskompatibilität unterstützt werden.
- Der Header muss das Feld: `Content-Type: application/json` enthalten. Wenn Sie Dateien allerdings im Body des `POST` Request hochladen möchten, muss der Content-Type des Headers auf `multipart/form-data` (d.h. `Content-Type: multipart/form-data`) gesetzt werden. Nähere Informationen dazu finden Sie im Abschnitt Hochladen von Dateien mit dem POST-Request.
- Der Body des Request muss im JSON-Format sein.
- Die zu verarbeitenden Dateien müssen sich auf dem Server befinden, d.h. die Dateien müssen entweder vor Absenden des Request auf den Server kopiert werden oder sie müssen zusammen mit dem POST Request hochgeladen werden. In diesem Fall muss der Content-Type des Message Headers auf `multipart/form-data` gesetzt werden. Nähere Informationen dazu finden Sie im Abschnitt Hochladen von Dateien mit dem POST-Request.

Der Request zur Überprüfung der Wohlgeformtheit einer XML-Datei würde im JSON-Format in etwa folgendermaßen aussehen:

```
{
   "command": "wfxml", "args": [ "file:///c:/Test/Report.xml" ]
}
```

Gültige Befehle, ihre Argumente und Optionen sind im Abschnitt zur Befehlszeile dokumentiert.

 © 2014 Altova GmbH

JSON-Syntax für HTTP `POST` Requests

```
{
  "command": "Command-Name",
  "options": {"opt1": "opt1-value", "opt2": "opt2-value"},
  "args"   : ["file:///c:/filename1", "file:///c:/filename2"]
}
```

- Der gesamte schwarze Text ist festgelegt und muss inkludiert werden. Dazu gehören alle geschweiften und eckigen Klammern, doppelten Anführungszeichen, Doppelpunkte und Kommas. Whitespaces können normalisiert werden.
- Bei Einträgen in blauer kursiver Schrift handelt es sich um Platzhalter. Sie stehen für Befehlsnamen, Optionen, Optionswerte und Argumentwerte. Eine Beschreibung der einzelnen Befehle finden Sie im Abschnitt zur Befehlszeile.
- Die Schlüssel `command` und `args` sind obligatorisch. Der Schlüssel `options` ist optional. Einige `options` Schlüssel haben Standardwerte, d.h. es müssen nur die Optionen angegeben werden, deren Standardwerte geändert werden müssen.
- Alle Strings müssen in doppelte Anführungszeichen gesetzt werden. Boolesche Werte und Zahlen dürfen keine Anführungszeichen haben. D.h.: {`"error-limit": "unlimited"`} und {`"error-limit": 1`} ist korrekt.
- Beachten Sie, dass es sich empfiehlt, anstelle von Dateipfaden Datei-URIs zu verwenden. Dafür werden Schrägstriche verwendet. Bei Verwendung von Windows-Dateipfaden werden umgekehrte Schrägstriche verwendet. Diese Windows Dateipfade müssen in JSON mit Escape versehen werden. (Das Escape-Zeichen ist der umgekehrte Schrägstrich: `"c:\\dir\\filename"`). Beachten Sie, dass URIs und Dateipfade Strings sind, die in Anführungszeichen gesetzt werden müssen.

Hier sehen Sie ein Beispiel mit Optionen. Beachten Sie, dass einige Optionen (wie `input` oder `xslt-version`) einen direkten Optionswert erhalten, während andere (wie `param`) ein Schlüsselwertpaar als Wert erhalten, wofür eine andere Syntax erforderlich ist.

```
{
   "command": "xslt",
   "args": [
           "file:///C:/Work/Test.xslt"
           ],
   "options": {
              "input": "file:///C:/Work/Test.xml",
              "xslt-version": 1,
              "param": {
                       "key": "myTestParam",
                       "value": "SomeParamValue"
```

```
                    },
                "output": "file:///C:/temp/out2.xml"
                }
}
```

Im unten gezeigten Beispiel sehen Sie eine dritte Art von Option: die eines Werte-Array (wie bei der Option `xsd` unten). In diesem Fall muss als Syntax die eines JSON Array verwendet werden.

```
{
   "command": "xsi",
   "args": [
           "file:///C:/Work/Test.xml"
           ],
   "options": {
              "xsd" : ["file:///C:/Work/File1.xsd", "file:///C:/Work/File2.xsd"]
              }
}
```

Hochladen von Dateien mit dem POST Request
Zu verarbeitende Dateien können im Body des `POST` Request hochgeladen werden. In diesem Fall muss der `POST` Request folgendermaßen erstellt werden.

Request Header
Setzen Sie im Request Header `Content-Type: multipart/form-data` und definieren Sie einen beliebigen String als Begrenzung. Hier sehen Sie einen Beispiel-Header:

```
Content-Type: multipart/form-data; boundary=---MyBoundary
```

Mit der Begrenzung (MyBoundary) werden die verschiedenen Formulardatenteile im Request Body voneinander abgegrenzt. (*siehe unten*).

Request Body: Message-Teil
Der Body des Request hat die folgenden Formulardatenteile, die durch den im Request Header definierten Begrenzungsstring (*siehe oben*) begrenzt sind:

- *Obligatorische Formulardatenteile:* **`msg`**, welches die angeforderte Verarbeitungsaktion definiert, und **`args`**, welches die Dateien enthält, die als das/die Argument(e) des im `msg` Formulardatenteil definierten Befehls hochzuladen ist/sind. *Siehe Codefragment unten.*
- *Optionaler Formulardatenteil:* Ein Formulardatenteil namens **`additional_files`**, welcher Dateien enthält, die von Dateien in den Formulardatenteilen `msg` oder `args` referenziert werden. Zusätzlich dazu können Formulardatenteile, die nach einer Option des Befehls benannt sind, ebenfalls hochzuladende Dateien enthalten.

 © 2014 Altova GmbH

Anmerkung: Alle hochzuladenden Dateien werden in einem einzigen virtuellen Verzeichnis angelegt.

Unten sehen Sie das Codefragment des Body eines `POST` Request. Die nummerierten Beschreibungen dazu finden Sie unterhalb. Der im Codefragment-Request gesendete Befehl hätte das folgende Befehlszeilenäquivalent:

```
raptorxml xsi First.xml Second.xml --xsd=Demo.xsd
```

Angefordert wird die Validierung von zwei XML-Dateien anhand eines Schemas. Der Body des Request würde in etwa folgendermaßen aussehen, wenn im Header `---PartBoundary` als Begrenzungsstring definiert wurde (*siehe Request Header* weiter oben).

```
---PartBoundary                                                          1
Content-Disposition: form-data; name="msg"
Content-Type: application/json

{"command": "xsi", "options": {}, "args": []}                           2

---PartBoundary                                                          3
Content-Disposition: attachment; filename="First.xml"; name="args"
Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                                 4
  <test xsi:noNamespaceSchemaLocation="Demo.xsd" xmlns:xsi="http://
  www.w3.org/2001/XMLSchema-instance">42</test>

---PartBoundary                                                          5
Content-Disposition: attachment; filename="Second.xml"; name="args"
Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                                 6
  <test xsi:noNamespaceSchemaLocation="Demo.xsd" xmlns:xsi="http://
  www.w3.org/2001/XMLSchema-instance">35</test>

---PartBoundary                                                          7
Content-Disposition: attachment; filename="Demo.xsd";
name="additional_files"
Content-Type: application/octet-stream

  <?xml version="1.0" encoding="UTF-8"?>                                 8
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
     <xs:element name="test" type="xs:int"/>
  </xs:schema>

---PartBoundary--                                                        9
```

1 Der Name der Begrenzungen für die Hauptformulardatenteile ist im Request Header deklariert. Der erste Formulardatenteil ist (in diesem Beispiel) **msg**. Beachten Sie, dass der Content-Type `application/json` ist.

2 Dies ist die Standardsyntax für HTTP POST Requests. Wenn `args` eine Referenz auf eine Datei enthält und wenn weitere Dateien hochgeladen werden, werden beide Dateigruppen an den Server übergeben.

3 Das erste Mitglied des `args` Array ist ein Dateianhang mit dem Namen `First.xml`.

4 Der Text der Datei `First.xml`. Er enthält eine Referenz auf ein Schema mit dem Namen `Demo.xsd`, welches ebenfalls - im Formulardatenteil **additional_files** - hochgeladen wird.

5 Das zweite Mitglied des `args` Array ist ein Anhang mit dem Namen `Second.xml`.

6 Der Text der Datei `Second.xml`. Auch dieser Teil enthält eine Referenz auf das Schema `Demo.xsd`. *Siehe Beschreibungstext 7.*

7 Der erste Zusatzdatenteil enthält die Metadaten für den Anhang `Demo.xsd` .

8 Der Text der Datei `Demo.xsd`.

9 Das Ende des Zusatzdatenteils `Demo.xsd` und des Formulardatenteils `additional_files`.

 © 2014 Altova GmbH

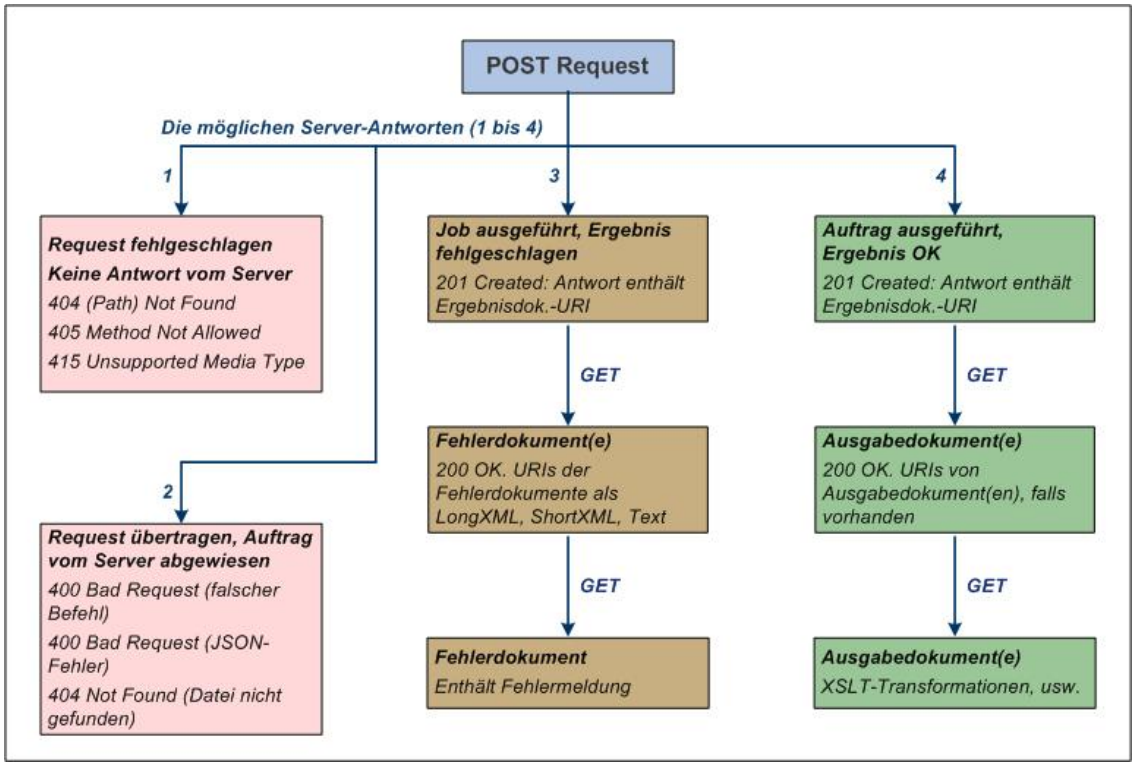
4.2.2 Server-Antwort auf den POST Request

In diesem Abschnitt werden folgende Schritte beschrieben:

- Übersicht über mögliche Server-Antworten
- Antwort: Request fehlgeschlagen, keine Antwort vom Server
- Antwort: Der Request wurde übertragen, doch der Auftrag wurde vom Server abgewiesen
- Antwort: Der Auftrag wurde ausgeführt (mit positivem oder negativem Ergebnis)

Wenn ein `POST` Request erfolgreich an den Server bereitgestellt wurde, wird der Auftrag in die Server-Warteschlange platziert. Der Server gibt eine `201 Created` Meldung und eine Ergebnisdokument-URI zurück. Der Auftrag wird zum frühestmöglichen Zeitpunkt verarbeitet. Falls in der Zwischenzeit das Ergebnisdokument angefordert wird, wird eine Meldung `"status": "Running"` zurückgegeben; der Client sollte das Dokument zu einem späteren Zeitpunkt wieder anfordern. Mit dem Status `Dispatched` wird angegeben, dass sich der Auftrag in der Server-Warteschlange befindet, aber noch nicht gestartet wurde.

Das Ergebnis des Auftrags (z.B. einer Validierungsanforderung) kann negativ (Validierung fehlgeschlagen) oder positiv (Validierung erfolgreich) sein. In beiden Fällen wird eine `201 Created` Meldung zurückgegeben und ein Ergebnisdokument wird generiert. Es kann auch sein, dass der `POST` Request nicht an den Server übertragen wurde (*Request fehlgeschlagen*) oder der Request zwar übertragen wurde, der Auftrag vom Server aber abgewiesen wurde (*Request übertragen, doch Auftrag abgewiesen*). Im nachstehenden Diagramm sind die verschiedenen möglichen Ergebnisse dargestellt.



Der `POST` Request des Client kann die folgenden möglichen Ergebnisse zur Folge haben:

Request fehlgeschlagen, keine Antwort vom Server
Wenn Requests nicht erfolgreich an den Server übertragen werden können, sind die häufigsten Fehler die folgenden:

Meldung	Erklärung
404 Not Found	Der richtige Pfad lautet: `http://localhost:8087/v1/queue/`
405 Method Not Allowed	Die angegebene Methode ist für diese Ressource unzulässig. Verwenden Sie die `POST` Methode.
415 Unsupported Media Type	Der Message Header sollte lauten: `Content-Type:application/json`.

Request übertragen, doch der Auftrag wurde vom Server abgewiesen
Wenn Requests erfolgreich an den Server übertragen wurden, kann der Server diese aus einem der folgenden Gründe abweisen:

Message	Erklärung
400 Bad Request (*bad cmd*)	Der RaptorXML Befehl ist falsch.
400 Bad Request (*json error*)	Der Request Body enthält einen JSON-Syntaxfehler.
404 File Not Found	Überprüfen Sie die Datei-URI (oder die Dateipfad)-Syntax aller im Befehl genannten Dateien.

Der Auftrag wurde ausgeführt (mit positivem oder negativem Ergebnis)
Wenn ein Auftrag (z.B. ein Validierungsauftrag) ausgeführt wird, kann das Ergebnis positiv (*OK*) oder negativ (*fehlgeschlagen*) sein. So kann z.B. das Ergebnis eines Validierungsauftrags positiv (*OK*) sein, wenn das zu validierende Dokument gültig ist und negativ (*fehlgeschlagen*), wenn das Dokument ungültig ist.

In beiden Fällen wurde der Auftrag ausgeführt, allerdings mit unterschiedlichen Ergebnissen. In beiden Fällen wird eine `201 Created` Meldung zurückgegeben, sobald der Auftrag erfolgreich in die Warteschlange aufgenommen wurde. In beiden Fällen wird auch eine Ergebnisdokument-URI an den HTTP-Client, von dem der Request stammt, zurückgegeben. (Möglicherweise wurde das Ergebnisdokument noch nicht erstellt, wenn mit der Verarbeitung des Auftrags noch nicht begonnen oder der Auftrag noch nicht fertig verarbeitet wurde.) Nachdem das Ergebnisdokument erstellt wurde, kann es mit einem HTTP `GET` Request abgerufen werden. Zusätzlich zum Ergebnisdokument können auch andere Dokumente generiert werden. Dazu gehören die folgenden:

 © 2014 Altova GmbH

- *Auftrag wurde mit dem Ergebnis 'Fehlgeschlagen' ausgeführt:* Ein Fehlerprotokoll wird in drei Formaten erstellt: Text, langes XML-Dokument und kurzes XML-Dokument. Die URIs dieser drei Dokumente werden im Ergebnisdokument (welches im JSON-Format ist) gesendet. Die URIs können in einem HTTP `GET` Request verwendet werden, um die Fehlerdokumente abzurufen.
- *Auftrag wurde mit dem Ergebnis 'OK' ausgeführt:* Der Auftrag wird erfolgreich ausgeführt und die Ausgabedokumente - wie z.B. die durch eine XSLT-Transformation erzeugte Ausgabe - werden erstellt. Wenn Ausgabedateien generiert wurden, werden ihre URIs im Ergebnisdokument, das im JSON-Format ist, gesendet. Die URIs können anschließend in einem HTTP `GET` Request verwendet werden, um die Ausgabedokumente abzurufen. Beachten Sie, dass nicht alle Aufträge Ausgabedateien haben; ein Beispiel dafür ist ein Validierungsauftrag. Außerdem kann es vorkommen, dass ein Auftrag mit dem Status '*OK*' fertig gestellt wurde, dass es aber Warnungen und/oder andere Meldungen gab, die in Fehlerdateien geschrieben wurden. In diesem Fall werden die URIs der Fehlerdateien (zusätzlich zu den Ausgabedokumenten) ebenfalls im Ergebnisdokument gesendet.

Eine Beschreibung dieser Dokumente und wie Sie diese aufrufen, finden Sie unter Abrufen des Ergebnisdokuments und Abrufen von Fehler-/Ausgabedokumenten.

4.2.3 Abrufen des Ergebnisdokuments

In diesem Abschnitt werden folgende Schritte beschrieben:

- die Ergebnisdokument-URI
- Abrufen des Ergebnisdokuments
  - Ergebnisdokument, das URIs von Fehlerdokumenten enthält
  - Ergebnisdokument, das URIs von Ausgabedokumenten enthält
  - Ergebnisdokument, das keine URI enthält
- Aufruf von im Ergebnisdokument aufgelisteten Fehler- und Ausgabedokumenten

---

Die Ergebnisdokument-URI
Bei jeder Erstellung eines Auftrags wird ein Ergebnisdokument erstellt, unabhängig davon, ob das Ergebnis eines Auftrags (z.B. einer Validierung) positiv (Dokument ist gültig) oder negativ (Dokument ungültig) ist. In beiden Fällen wird eine `201 Created` Meldung zurückgegeben. Diese Meldung ist im JSON-Format und enthält eine relative URI des Ergebnisdokuments. Das JSON-Fragment sieht in etwa folgendermaßen aus:

```
{
  "result": "/v1/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB",
  "jobid": "E6C4262D-8ADB-49CB-8693-990DF79EABEB"
}
```

Das Objekt `result` enthält die relative URI des Ergebnisdokuments. Die URI ist relativ zur Serveradresse. Wenn die Serveradresse z.B. `http://localhost:8087/` (die Adresse in der Anfangskonfiguration) ist, so lautet die erweiterte URI des im Codefragment oben angegebenen Ergebnisdokuments:

```
http://localhost:8087/v1/results/E6C4262D-8ADB-49CB-8693-990DF79EABEB
```

Anmerkung: Die richtige Versionsnummer `/vN` ist diejenige, die der Server zurückgibt (und nicht notwendigerweise die in der Dokumentation verwendete). Die Nummer, die der Server zurückgibt, ist die Versionsnummer der aktuellen HTTP-Schnittstelle. Frühere Versionsnummern stehen für ältere Versionen der HTTP-Schnittstelle, die jedoch weiterhin aus Gründen der Rückwärtskompatibilität unterstützt werden.

Abrufen des Ergebnisdokuments
Um das Ergebnisdokument abzurufen, senden Sie die erweiterte URI des Dokuments (*siehe oben*) in einem HTTP `GET` Request. Daraufhin wird das Ergebnisdokument zurückgegeben. Es könnte eine der unten beschriebenen allgemeinen Arten sein.

Anmerkung: Wenn ein Auftrag erfolgreich in die Server-Warteschlange gestellt wurde, wird die URI des Ergebnisdokuments zurückgegeben. Wenn der Client das Ergebnis anfordert, bevor der Auftrag gestartet wurde (weil er sich noch in der Warteschlange befindet) wird die Meldung `"status": "Dispatched"` zurückgegeben. Wenn der Auftrag gestartet, aber noch nicht fertig gestellt wurde (weil es sich z.B. um einen großen Auftrag handelt), wird

die Meldung `"status": "Running"` zurückgegeben. In beiden Fällen sollte der Client einige Zeit warten, bevor er das Ergebnisdokument erneut anfordert.

Anmerkung: In den Beispieldokumenten unten wird immer davon ausgegangen, dass der Zugriff des Client eingeschränkt ist. Daher wird angenommen, dass Fehlerdokumente, Meldungsdokumente und Ausgabedokumente im entsprechenden Auftragsverzeichnis auf dem Server gespeichert werden. Die URIs für diese Dokumente im Ergebnisdokument sind daher alle relative URIs. Keine davon ist eine Datei-URI (wie dies bei unbeschränktem Client-Zugriff der Fall wäre). Nähere Informationen zu diesen URIs finden Sie im Abschnitt Abrufen von Fehler-/Ausgabedokumenten.

Ergebnisdokument, das URIs von Fehlerdokumenten enthält

Wenn der angeforderte Auftrag mit dem Status *Fehlgeschlagen* beendet wurde, hat der Auftrag ein negatives Ergebnis zurückgegeben. So wurde z.B. bei einem Validierungsauftrag das Ergebnis: "Dokument ungültig" zurückgegeben. Die bei der Ausführung des Auftrags ausgegebenen Fehler werden in Fehlerprotokollen gespeichert, die in drei Formaten erstellt werden: (i) Text, (ii) long-XML (detailliertes Fehlerprotokoll) und (iii) short-XML (weniger ausführliches Fehlerprotokoll). Siehe JSON-Codefragment unten.

```
{
  "jobid": "6B4EE31B-FAC9-4834-B50A-582FABF47B58",
  "state": "Failed",
  "error":
  {
    "text": "/v1/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/error.txt",
    "longxml": "/v1/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/
long.xml",
    "shortxml": "/v1/results/6B4EE31B-FAC9-4834-B50A-582FABF47B58/error/
short.xml"
  },
  "jobs":
  [
    {
      "file": "file:///c:/Test/ExpReport.xml",
      "jobid": "20008201-219F-4790-BB59-C091C276FED2",
      "output":
      {
      },
      "state": "Failed",
      "error":
      {
        "text": "/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/
error.txt",
        "longxml": "/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/
long.xml",
        "shortxml": "/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/
short.xml"
      }
    }
  ]
}
```

Beachten Sie bitte die folgenden Punkte:

- Aufträge haben Unteraufträge.
- Fehler, die auf Unterauftragsebene entstehen, setzen sich bis zum Auftrag der obersten Ebene fort. Der Status des Auftrags der obersten Ebene ist nur dann *OK,* wenn alle seine

Unteraufträge ebenfalls den Status *OK* haben.
- Jeder Auftrag oder Unterauftrag hat sein eigenes Fehlerprotokoll.
- Fehlerprotokolle enthalten Warnungsprotokolle, d.h. auch wenn ein Auftrag mit dem Status *OK* beendet wird, kann er URIs von Fehlerdateien enthalten.
- Die URIs von Fehlerdateien sind relativ zur Serveradresse (*siehe oben*).

Ergebnisdokument, das URIs von Ausgabedokumenten enthält
Wenn der angeforderte Auftrag mit dem Status *OK* beendet wurde, wurde vom Auftrag ein positives Ergebnis zurückgegeben. So wurde z.B. bei einem Validierungsauftrag das Ergebnis: "Dokument gültig" zurückgegeben. Wenn beim Auftrag ein Ausgabedokument - z.B. das Ergebnis einer XSLT-Transformation - zurückgegeben wurde, so wird die URI des Ausgabedokuments zurückgegeben. Siehe JSON-Codefragment unten.

```
{
  "jobid": "5E47A3E9-D229-42F9-83B4-CC11F8366466",
  "state": "OK",
  "error":
  {
  },
  "jobs":
  [
    {
      "file": "file:///c:/Test/SimpleExample.xml",
      "jobid": "D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8",
      "output":
      {
        "xslt-output-file":
        [
          "/v1/results/D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8/output/test.html"
        ]
      },
      "state": "OK",
      "output-mapping":
      {
        "/v1/results/D34B5684-C6FF-4A7A-BF35-EBB9A8A8C2C8/output/1": "file:///
        c:/temp/test.html"
      },
      "error":
      {
      }
    }
  ]
}
```

Beachten Sie die folgenden Punkte:

- Die Ausgabedatei wird im Ordner `output` des Auftrags erstellt. Sie können zum Aufrufen der Datei die relative URI verwenden.
- Die URIs der Ausgabedateien sind relativ zur Serveradresse (*siehe oben*).
- Das Datenelement `output-mapping` mappt das Ausgabedokument im Auftragsverzeichnis auf dem Server auf den durch den Client in der Auftragsanforderung angegebenen Ordner. Beachten Sie, dass nur Ausgabedokumente, die vom Client in der Auftragsanforderung definiert sind, ein Mapping haben; Dateien im Zusammenhang mit dem Auftrag, die vom Server generiert werden (wie z.B. Fehlerdateien), haben kein Mapping.

 © 2014 Altova GmbH

Ergebnisdokument, das keine URI enthält
Wenn der angeforderte Auftrag mit dem Status *OK* beendet wurde, hat der Auftrag ein positives Ergebnis zurückgegeben. So wurde z.B. bei einem Validierungsauftrag das Ergebnis: "Dokument gültig" zurückgegeben. Bei einigen Aufträgen - z.B. bei der Validierung oder der Wohlgeformtheitsprüfung - wird kein Ausgabedokument erzeugt. Wenn ein Auftrag dieser Art mit dem Status *OK* beendet wird, hat das Ergebnisdokument weder die URI eines Ausgabedokuments noch die URI eines Fehlerprotokolls. Siehe JSON-Codefragment unten.

```
{
  "jobid": "3FC8B90E-A2E5-427B-B9E9-27CB7BB6B405",
  "state": "OK",
  "error":
  {
  },
  "jobs":
  [
    {
      "file": "file:///c:/Test/SimpleExample.xml",
      "jobid": "532F14A9-F9F8-4FED-BCDA-16A17A848FEA",
      "output":
      {
      },
      "state": "OK",
      "error":
      {
      }
    }
  ]
}
```

Beachten Sie die folgenden Punkte:

- Sowohl das Ausgabedokument als auch das Fehlerprotokoll des Unterauftrags im Codefragment oben sind leer.
- Ein Auftrag, der mit dem Status *OK* beendet wurde, kann trotzdem Warnungen oder andere Meldungen enthalten, die in Fehlerdateien protokolliert sind. In diesem Fall enthält das Ergebnisdokument URIs von Fehlerdateien, obwohl der Auftrag mit dem Status *OK* beendet wurde.

Aufrufen von im Ergebnisdokument aufgelisteten Fehler- und Ausgabedokumenten
Fehler- und Ergebnisdokumente können mit HTTP `GET` Requests aufgerufen werden. Eine Beschreibung dazu finden Sie im nächsten Abschnitt Abrufen von Fehler-/Meldungs-/Ausgabedokumenten.

4.2.4 Abrufen von Fehler-/Meldungs-/Ausgabedokumenten

Ein Ergebnisdokument kann die Datei-URIs oder relativen URIs von Fehlerdokumenten, Meldungsdokumenten (wie z.B. Logs) und/oder Ausgabedokumenten enthalten. (In manchen Fällen enthält das Ergebnisdokument keine URI.) Im Folgenden sind die verschiedenen URI-Arten beschrieben.

Um diese Dokumente über HTTP aufrufen zu können, gehen Sie folgendermaßen vor:

1. Erweitern Sie die relative URI der Datei im Ergebnisdokument zu ihrer absoluten URI.
2. Verwenden Sie die erweiterte URI in einem HTTP GET Request, um die Datei aufzurufen.

URIs von Fehler-/Meldungs-/Ausgabedokumenten (im Ergebnisdokument)
Das Ergebnisdokument enthält URIs von Fehler-, Meldungs- und/oder Ausgabedokumenten. Fehler- und Meldungsdokumente sind Dokumente, die vom Server im Zusammenhang mit einem Auftrag generiert werden; sie werden immer im Auftragsverzeichnis auf dem Server gespeichert. Ausgabedokumente (wie z.B. die Ausgabe von XSLT-Transformationen) können in einem der folgenden Ordner gespeichert werden:

- unter jedem Dateipfad, auf den der Server Zugriff hat. Damit Ausgabedateien in jedem beliebigen Ordner gespeichert werden können, muss der Server für den unbeschränkten Client-Zugriff konfiguriert sein (Standardeinstellung).
- im Auftragsverzeichnis auf dem Server. Der Server ist so konfiguriert, dass er nur eingeschränkten Zugriff von Clients gestattet.

Wenn ein Client verlangt, dass eine Ausgabedatei erstellt wird, hängt der Pfad, unter dem die Ausgabedatei gespeichert wird, von der Option `server.unrestricted-filesystem-access` der Server-Konfigurationsdatei ab.

- Wenn unbeschränkter Zugriff besteht, wird die Datei in dem vom Client angegebenen Ordner gespeichert. Die für das Dokument zurückgegebene URI ist in diesem Fall eine Datei-URI.
- Wenn der Zugriff eingeschränkt ist, wird die Datei im Auftragsverzeichnis gespeichert. Ihre URI ist dann eine relative URI. Außerdem gibt es ein Mapping dieser relativen URI auf die vom Client angegebene URL. (*Siehe Codefragment in Ergebnisdokument, das URIs von Ausgabedokumenten enthält*)

Zusammenfassend gibt es die folgenden URI-Arten:

Datei-URI von Fehler-/Meldungsdokumenten
Diese Dokumente werden im Auftragsverzeichnis auf dem Server gespeichert. Datei-URIs haben die folgende Form:

```
Datei:///<Root-Ausgabe-Verz>/AUFTRAGSID/meldung.doc
```

Datei-URIs von Ausgabedokumenten
Diese Dokumente können in jedem beliebigen Ordner gespeichert werden. Datei-URIs haben die folgenden Form:

```
Datei:///<Pfad-zur-Datei>/ausgabe.doc
```

HTTP-URI von Fehler-/Meldungs-/Ausgabedokumenten

Diese Dokumente werden im Auftragsverzeichnis auf dem Server gespeichert. URIs sind relativ zur Server-Adresse und müssen zur vollständigen HTTP-URI erweitert werden. Die relative URI hat die folgende Form:

`/vN/Ergebnisse/AUFTRAGSID/Fehler/Fehler.txt`	für Fehlerdokumente
`/vN/Ergebnisse/AUFTRAGSID/Ausgabe/verbose.log`	für Meldungsdokumente
`/vN/Ergebnisse/AUFTRAGSID/Ausgabe/1`	für Ausgabedokumente

Bei Ausgabedokumenten werden Ausgabe-Mappings geliefert (*siehe Beispielcode*). Damit werden die einzelnen Ausgabedokument-URIs im Ergebnisdokument auf das entsprechende Dokument im Client Request gemappt.

Erweitern der relativen URI
Erweitern Sie die relative URI im Ergebnisdokument zu einer absoluten HTTP-URI, indem Sie der relativen URI die Serveradresse voranstellen. Wenn die Serveradresse z.B. folgendermaßen lautet:

`http://localhost:8087/` (die Adresse in der Anfangskonfiguration)

und wenn die relative URI einer Fehlerdatei im Ergebnisdokument folgendermaßen lautet:

`/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt`

so ist die erweiterte absolute Adresse:

`http://localhost:8087/v1/results/20008201-219F-4790-BB59-C091C276FED2/error/error.txt`

Nähere Informationen im Zusammenhang damit finden Sie in den Abschnitten: Konfigurieren des Servers und Abrufen des Ergebnisdokuments.

Verwenden eines HTTP `GET` Request zum Aufrufen der Datei
Verwenden Sie die erweiterte URI in einem HTTP `GET` Request, um die gewünschte Datei zu erhalten. RaptorXML Server gibt daraufhin das angeforderte Dokument zurück.

4.2.5 Freigeben von Serverressourcen nach der Verarbeitung

RaptorXML Server speichert die Ergebnisdokumentdatei, temporäre Dateien und Fehler- und Ausgabedokumente zu einem verarbeiteten Auftrag auf der Festplatte. Diese Dateien können auf zwei Arten gelöscht werden:

- Durch Angeben der URI des Ergebnisdokuments mit der HTTP `DELETE` Methode. Dadurch werden alle Dateien im Zusammenhang mit dem Auftrag, der durch die gesendete Ergebnisdokument-URI referenziert wird, einschließlich aller Fehler- und Ausgabedokumente gelöscht.
- Durch manuelles Löschen einzelner Dateien auf dem Server durch einen Administrator.

Die Struktur der URI, die mit der HTTP `DELETE` Methode verwendet wird, entspricht der unten gezeigten. Beachten Sie, dass die vollständige URI aus der Serveradresse plus der relativen URI des Ergebnisdokuments besteht.

HTTP-Methode	URI
DELETE	http://localhost:8087/v1/result/D405A84A-AB96-482A-96E7-4399885FAB0F

Zur Angabe des Ausgabeverzeichnisses eines Auftrags auf der Festplatte, konstruieren Sie die URI folgendermaßen:

```
[<server.output-root-dir> siehe Server-Konfigurationsdatei] + [Auftragsid]
```

Anmerkung: Da oft zahlreiche Fehler- und Ausgabedokumente erstellt werden, empfiehlt es sich, die Verwendung des Festplattenspeichers zu überwachen und je nach Umgebung und Ihren Anforderungen regelmäßig Löschungen vorzunehmen.

© 2014 Altova GmbH

Kapitel 5

Python-Schnittstelle

5 Python-Schnittstelle

Über die Python-Schnittstelle von RaptorXML Server können Daten in XML-Dokumenten und XML-Schema-Dokumenten über die Python APIs für XML und XSD aufgerufen und verarbeitet werden. Welche Daten in den Quelldokumenten verarbeitet werden sollen und wie diese Daten verarbeitet werden sollen, ist im Python-Skript, das an RaptorXML Server übergeben wird, definiert.

Die Python-APIs
Die Python-APIs (für XML und XSD) bieten Zugriff auf die Metainformationen, Strukturinformationen und Daten in XML- und XSD-Dokumenten. Somit können Python-Skripts erstellt werden, die über die APIs auf Dokumentinformationen zugreifen und diese verarbeiten. So kann z.B. ein Python-Skript an RaptorXML Server übergeben werden, das Daten aus einem XML-Dokument in eine Datenbank oder eine CSV-Datei schreibt.

Die Python-APIs sind in den folgenden Abschnitten beschrieben:

- Python-XML-API
- Python-XSD-API

Python-Skripts
Ein vom Benutzer erstelltes Python-Skript wird mit dem Parameter **`--script`** der folgenden Befehle übergeben:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`

Diese Befehle, die Python-Skripts aufrufen, können sowohl über die Befehlszeilenschnittstelle (CLI) als auch über die HTTP-Schnittstelle verwendet werden. Die Verwendung von Python-Skripts mit den Python-APIs von RaptorXML Server ist in den Abschnitten Erstellen von Python-Skripts und Ausführen von Python-Skripts beschrieben.

Python-Skripts sicher machen
Wenn ein Python-Skript in einem Befehl über HTTP an RaptorXML Server adressiert ist, funktioniert das Skript nur, wenn es sich im vertrauenswürdigen Verzeichnis befindet. Das Skript wird vom vertrauenswürdigen Verzeichnis aus ausgeführt. Wenn Sie ein Python-Skript aus einem anderen Verzeichnis definieren, wird ein Fehler ausgegeben. Das vertrauenswürdige Verzeichnis wird in der `server.script-root-dir` Einstellung der Serverkonfigurationsdatei definiert. Wenn Sie Python-Skripts verwenden möchten, **muss** ein vertrauenswürdiges Verzeichnis definiert werden. Stellen Sie sicher, dass alle Python-Skripts, die verwendet werden sollen, in diesem Verzeichnis gespeichert werden.

Zwar werden alle vom Server für HTTP-Auftragsanforderungen generierten Ausgabedateien in das

 © 2014 Altova GmbH

Auftragsausgabeverzeichnis (ein Unterverzeichnis von `output-root-directory`) geschrieben, doch gilt diese Einschränkung nicht für Python-Skripts, die in jeden Ordner geschrieben werden können. Der Server-Administrator muss die Python-Skripts im vertrauenswürdigen Verzeichnis auf potentielle Schwachstellen überprüfen.

5.1 Erstellen von Python-Skripts

In diesem Abschnitt werden folgenden Punkte beschrieben:

- Python-Version
- Speichern von Python-Skripts
- Übergeben eines Python-Skripts an RaptorXML Server
- Python-Eintrittspunktfunktionen
- Vereinfachte Struktur eines Python-Skripts
- Die Python-Eintrittspunktfunktion im Detail

Python-Version
Vom Benutzer erstellte Python-Skripts müssen mindestens mit **Python 3.3.1** kompatibel sein.

Python-Skripts sicher machen
Wenn ein Python-Skript in einem Befehl über HTTP an RaptorXML Server adressiert ist, funktioniert das Skript nur, wenn es sich im vertrauenswürdigen Verzeichnis befindet. Das Skript wird vom vertrauenswürdigen Verzeichnis aus ausgeführt. Wenn Sie ein Python-Skript aus einem anderen Verzeichnis definieren, wird ein Fehler ausgegeben. Das vertrauenswürdige Verzeichnis wird in der `server.script-root-dir` Einstellung der Serverkonfigurationsdatei definiert. Wenn Sie Python-Skripts verwenden möchten, **muss** ein vertrauenswürdiges Verzeichnis definiert werden. Stellen Sie sicher, dass alle Python-Skripts, die verwendet werden sollen, in diesem Verzeichnis gespeichert werden.

Zwar werden alle vom Server für HTTP-Auftragsanforderungen generierten Ausgabedateien in das Auftragsausgabeverzeichnis (ein Unterverzeichnis von `output-root-directory`) geschrieben, doch gilt diese Einschränkung nicht für Python-Skripts, die in jeden Ordner geschrieben werden können. Der Server-Administrator muss die Python-Skripts im vertrauenswürdigen Verzeichnis auf potentielle Schwachstellen überprüfen.

Übergeben eines Python-Skripts an RaptorXML Server
Ein Python-Skript wird mit dem **`--script`**-Parameter der folgenden Befehle übergeben:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`

Diese Befehle können über die Befehlszeilenschnittstelle oder die HTTP-Schnittstelle verwendet werden. Ein Beispiel dazu finden Sie im Abschnitt Ausführen von Python-Skripts.

 © 2014 Altova GmbH

Python-Eintrittspunktfunktionen
Die Befehle, die Zugriff auf die Python-Schnittstelle gestatten (*siehe Liste oben*), sind Validierungsbefehle und das Python-Skript wird nur dann ausgeführt, wenn die mit dem Befehl übergebenen Dateien gültig sind. Nachdem die Validierung erfolgreich ausgeführt wurde, ruft RaptorXML Server je nachdem, welcher Befehl ausgeführt wurde, eine bestimmte Funktion auf. Die aufgerufene Funktion (*siehe Tabelle unten*) muss daher im Python-Skript definiert sein. Sie muss mit zwei Parametern definiert sein: der erste ist das Auftragsobjekt, der zweite ist abhängig davon, welcher Befehl ausgeführt wurde (*siehe Tabelle*).

Befehl	Von RaptorXML Server aufgerufene Funktion
valxml-withxsd (xsi)	**on_xsi_valid**(job,instance)
valxsd (xsd)	**on_xsd_valid**(job,schema)

Vereinfachte Struktur des Python-Skripts
Ein Python-Skript, über das auf die Python-Schnittstelle zugegriffen wird, hat die folgende grundlegende Struktur. Beachten Sie, wie die Python-Eintrittspunktfunktion definiert ist.

```
1   import os
    from altova import xml, xsd

2   def on_xsi_valid(job,instance):
      filename = os.path.join(job.output_dir,'script_out.txt')
      job.append_output_filename(filename)
      f = open(filename,'w')
      f.write(str(type(job))+'\n')
      f.write(str(job)+'\n')
      f.write(job.output_dir+'\n')
      f.close()
      filename2 = os.path.join(job.output_dir,'script_out2.txt')
      job.append_output_filename(filename2)
      f2 = open(filename2,'w')
      print_instance(f2,instance)
      f2.close()

3   CodeBlock-1
       ...
    CodeBlock-N
```

Beschreibung der Struktur des obigen Skripts:

1. Importiert das vordefinierte `os` Modul von Python und anschließend das **`xml`**, **`xsd`**-Modul der Altova-Bibliothek.
2. Die **Python-Eintrittspunktfunktion** (*siehe unten*). Dabei könnte es sich um eine der folgenden handeln:
   `on_xsi_valid``(job,instance)`, **`on_xsd_valid`**`(job,schema)`, **`on_dts_valid`**`(job,dts)`, **`on_xbrl_valid`**`(job,instance)`.

3 Zusätzliche Codeblöcke, von denen jeder Funktionsdefinitionen oder anderen Code enthält.

Die Python-Eintrittspunktfunktion im Detail
In diesem Abschnitt werden anhand der folgenden Eintrittspunktfunktionsdefinition wichtige Punkte zur Python-Eintrittspunktfunktion erläutert.

```
def on_xsi_valid(job,instance):
  filename = os.path.join(job.output_dir,'script_out.txt')
  job.append_output_filename(filename)
  f = open(filename,'w')
  f.write(str(type(job))+'\n')
  f.write(str(job)+'\n')
  f.write(job.output_dir+'\n')
  f.close()
  filename2 = os.path.join(job.output_dir,'script_out2.txt')
  job.append_output_filename(filename2)
  f2 = open(filename2,'w')
  print_instance(f2,instance)
  f2.close
```

- Die Zeile `def on_xsi_valid(job,instance):` beginnt den Funktionsdefinitionsblock.
- Die Funktion hat den Namen **`on_xsi_valid`**`(job,instance)` und erhält zwei Argumente: `job` und `instance`.
- Dies ist die Funktion, die aufgerufen wird, nachdem RaptorXML Server den Befehl `valxml-withxsd (xsi)` erfolgreich ausgeführt und die angegebene(n) XML-Datei(en) als gültig validiert hat.
- Die Werte der Argumente `job` und `instance` werden von RaptorXML Server bereitgestellt.
- Der Wert der Variablen `filename` wird mit Hilfe von `job.output_dir` konstruiert. Der Wert von `job.output_dir` ist bei Verwendung der HTTP-Schnittstelle in der Server-Konfigurationsdatei definiert. Bei Verwendung des CLI ist er im Arbeitsverzeichnis definiert.
- Die Funktion `job.append_output_filename` hängt einen Dateinamen an die Auftragsausgabe an.

 © 2014 Altova GmbH

5.2 Ausführen von Python-Skripts

Python-Skripts werden durch Angabe der URL des Skripts als Wert der Option `--script` an RaptorXML Server übergeben. Die Option `--script` wird für die folgenden Befehle unterstützt:

- `valxml-withxsd (xsi)`
- `valxsd (xsd)`

Diese Befehle können über die Befehlszeilenschnittstelle oder die HTTP-Schnittstelle verwendet werden.

Beispiele
Hier finden Sie Beispiele für die Verwendung mit den verschiedenen Befehlen:

- `raptorxml xsi --script=xml.py --streaming=false c:\HasXSDRef.xml`
- `raptorxml xsd --script=xsd.py c:\Test.xsd`

Anmerkung: Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd` verwenden, stellen Sie sicher, dass `streaming=false` definiert ist. Andernfalls wird eine Warnung zurückgegeben, dass das Skript nicht ausgeführt wurde.

Starten des Skripts
Nachdem der Befehl erfolgreich übergeben wurde und die Datei(en) als gültig validiert wurden, ruft RaptorXML Server die entsprechende Python-Eintrittspunktfunktion für den gerade ausgeführten Befehl auf und übergibt ihr die Werte der beiden Argumente der Funktion. Wenn die Eintrittspunktfunktion sich im Skript befindet, das mit dem Parameter `--script` übergeben wurde, wird mit der Ausführung des Skripts begonnen.

5.3 Beispielskript 01: Verarbeiten von XML

Dieses Python-Skript verarbeitet Daten in der Datei **NanonullOrg.xml** (aus dem Ordner `examples` des RaptorXML-Applikationsordners) und erstellt ein Ausgabedokument namens **summary.html**, das eine Tabelle enthält, in der die Gesamtanzahl der Aktien, die sich im Besitz der Angestellten jeder einzelnen Abteilung befinden, summiert ist.

Das Skript wird mit einem Befehl, wie dem folgenden an die Befehlszeilenschnittstelle übergeben:

```
raptorxml xsi --streaming=false --script=sharesummary.py NanonullOrg.xml
```

Dieser Abschnitt enthält die folgenden Codefragmente:

- das mit Anmerkungen versehene Python-Skript
- das vom Skript erzeugte Ergebnisdokument

© 2014 Altova GmbH

5.3.1 Skriptcode

Das folgende mit Anmerkungen versehene Python-Skript verarbeitet Daten in der Datei **NanonullOrg.xml** (aus dem Ordner `examples` des RaptorXML-Applikationsordners) und erstellt ein Ausgabedokument namens **summary.html**. Das Ausgabedokument enthält eine Tabelle, in der die Gesamtanzahl der Aktien, die sich im Besitz der Angestellten jeder einzelnen Abteilung befinden, summiert ist.

Das Skript kann mit einem Befehl wie dem folgenden an die Befehlszeilenschnittstelle übergeben werden:

```
raptorxml xsi --streaming=false --script=sharesummary.py NanonullOrg.xml
```

Anmerkung: Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd | xsi` verwenden, stellen Sie sicher, dass `streaming=false` definiert ist. Andernfalls wird eine Warnung zurückgegeben, dass das Skript nicht ausgeführt wurde.

Dateiname: sharesummary.py

```
import os
from altova import xml

def getElemTextValue(elem):
    """Returns the text content of an XML element"""

    text = ''
    for child in elem.children:
        if isinstance(child,xml.Character):
            text += child.character_code
    return text


def getChildElemsWithName(elemParent,name):
    """Returns a list of all child elements with the given name"""

    elems = []
    for child in elemParent.children:
        if isinstance(child,xml.Element) and child.local_name == name:
            elems.append(child)
    return elems


def getDepartmentName(elemDepartment):
    """Returns the name of the department specified in the <Name> element"""

    return getElemTextValue(getChildElemsWithName(elemDepartment,'Name')[0])


def getDepartmentTotalShares(elemDepartment):
    """Returns the number of shares held by each person in that department"""

    # Initialize total shares to 0
    totalShares = 0
```

```
    # Sum the shares of each <Person> within the deparment
    for elemPerson in getChildElemsWithName(elemDepartment,'Person'):
        elemShares = getChildElemsWithName(elemPerson,'Shares')
        # <Shares> element is optional, thus we need to check for its existence
        if len(elemShares):
            # Get the value of the <Shares> element, convert it to an integer and
            add it to the total sum
            totalShares += int(getElemTextValue(elemShares[0]))
    return totalShares


def calcSharesPerDepartment(instance):
    """Return a map containing the number of shares held by the persons in each
    department"""

    # Get XML root element
    elemOrgChart  = instance.document.document_element
    # Check if the root element is <OrgChart>
    if not elemOrgChart or elemOrgChart.local_name != 'OrgChart' or
    elemOrgChart.namespace_name != 'http://www.xmlspy.com/schemas/orgchart':
        # Otherwise raise error
        raise Error('This script must be used with a valid OrgChart instance!')

    mapSharesPerDepartment = {}
    # Go through each <Department> in each <Office> and set the number of shares
    held by each person in that department
    for elemOffice in getChildElemsWithName(elemOrgChart,'Office'):
        for elemDepartment in getChildElemsWithName(elemOffice,'Department'):
            mapSharesPerDepartment[getDepartmentName(elemDepartment)] =
            getDepartmentTotalShares(elemDepartment)
    return mapSharesPerDepartment


def writeSummary(mapSharesPerDepartment,filename):
    """Write a summary containing the number of shares for each department to the
    give filename"""

    # Open file for writing
    f = open(filename,'w')
    f.write('<html><title>Summary</title><body><table border="1">\n')
    f.write('<tr><th>Department</th><th>Shares</th></tr>\n')
    # Generate a table row for each department with the deparment's name and its
    total number of shares
    for name,shares in sorted(mapSharesPerDepartment.items()):
        f.write('<tr><td>%s</td><td>%d</td></tr>\n'%(name,shares))
    f.write('</table></body></html>\n')
    # Close file
    f.close()


def on_xsi_valid(job,instance):
    """This method will be automatically called by RaptorXML after successful
    validation of the XML instance"""

    # Create a 'summary.html' file in the job's ouptut directory (when run from
    the CLI this will be the current working directory)
```

 © 2014 Altova GmbH

```
filename = os.path.join(job.output_dir,'summary.html')
# Calculate the number of shares per department and write a summary to
'summary.html'
writeSummary(calcSharesPerDepartment(instance),filename)
# Register the newly generated 'summary.html' output file
job.append_output_filename(filename)
```

5.3.2 Ergebnisdokument

Im Folgenden finden Sie den vom Python-Skript sharesummary.py. erzeugten Code des Dokuments summary.html.

Dateiname: summary.html

```
<html><title>Summary</title><body>
  <table border="1">
    <tr><th>Department</th><th>Shares</th></tr>
    <tr><td>Administration</td><td>2500</td></tr>
    <tr><td>Engineering</td><td>5500</td></tr>
    <tr><td>IT & Technical Support</td><td>1750</td></tr>
    <tr><td>Marketing</td><td>3000</td></tr>
    <tr><td>Research & Development</td><td>5500</td></tr>
  </table>
</body></html>
```


© 2014 Altova GmbH

5.4 Beispielskript 02: XML neu formatieren

Das Python-Skript in diesem Beispiel formatiert die XML-Datei in der Datei **NanonullOrg.xml** (aus dem Ordner `examples` des RaptorXML-Applikationsordners) neu. Jedes Element wird mit Tabulatoren eingerückt und jedes Attribut wird in eine separate Zeile gesetzt (wodurch der Vergleich mit einem Vergleichstool übersichtlicher wird). Das Ausgabedokument hat den Namen **output.xml**.

Das Skript wird mit einem Befehl wie dem folgenden an die Befehlszeilenschnittstelle übergeben:

```
raptorxml xsi --streaming=false --script=reformat.py NanonullOrg.xml
```

Dieser Abschnitt enthält die folgenden Codefragmente:

- das mit Anmerkungen versehene Python-Skript
- das vom Skript erzeugte Ergebnisdokument

5.4.1 Skriptcode

Das folgende mit Anmerkungen versehene Python-Skript (`reformat.py`) formatiert die XML-Datei **`NanonullOrg.xml`** (aus dem Ordner `examples` des RaptorXML-Applikationsordners). Jedes Element wird mit Tabulatoren eingerückt und jedes Attribut wird in eine separate Zeile gesetzt (wodurch der Vergleich mit einem Vergleichstool übersichtlicher wird). Das Ausgabedokument hat den Namen **`output.xml`**.

Das Skript wird mit einem Befehl wie dem folgenden an die Befehlszeilenschnittstelle übergeben:

```
raptorxml xsi --streaming=false --script=reformat.py NanonullOrg.xml
```

Anmerkung: Wenn Sie die Option `--script` mit dem Befehl `valxml-withxsd | xsi` verwenden, stellen Sie sicher, dass `streaming=false` definiert ist. Andernfalls wird eine Warnung zurückgegeben, dass das Skript nicht ausgeführt wurde.

Dateiname: reformat.py

```
import os
from altova import xml, xsd

def writeCharacter(f,char,depth):
    """Output XML for the charater node"""

    # Ignore text nodes containing only whitespace characters
    if not char.element_content_whitespace:
        # Write the text content
        f.write("\t"*depth + char.character_code+'\n')


def writeComment(f,comment,depth):
    """Output XML for the comment node"""

    # Write the comment
    f.write("\t"*depth + '<!-- '+comment.content+'-->\n')


def writeAttribute(f,attr,depth):
    """Output XML for the attribute node (on a separate line)"""

    # Look up prefix for the namespace in the inscope namespace map
    prefix = None
    if attr.namespace_name:
        inscope = {}
        for namespace in attr.owner_element.inscope_namespaces:
            inscope[namespace.namespace_name] = namespace.prefix
        prefix = inscope[attr.namespace_name]
        if prefix:
            prefix += ':'
    if not prefix:
        prefix = ''
```

 © 2014 Altova GmbH

```
    # Write the attribute with its value
    f.write("\t"*depth + "@"+prefix+attr.local_name+"=\""+attr.normalized_value
    +"\"\n")


def writeNSAttribute(f,attr,depth):
    """Output XML for the namespace attribute node (on a separate line)"""

    prefix = ""
    if attr.local_name != 'xmlns':
        prefix = 'xmlns:'

    # Write the namespace attribute with its value
    f.write("\t"*depth + "@"+prefix+attr.local_name+"=\""+attr.normalized_value
    +"\"\n")


def writeChildren(f,elem,depth):
    """Output XML for all the child nodes (indented by the given depth)"""

    # Iterate over all child nodes
    for child in elem.children:
        if isinstance(child,xml.Element):
            writeElement(f,child,depth)
        elif isinstance(child,xml.Comment):
            writeComment(f,child,depth)
        elif isinstance(child,xml.Character):
            writeCharacter(f,child,depth)


def writeElement(f,elem,depth):
    """Output XML for the element node with all its child nodes (indented by the
    given depth)"""

    # Look up prefix for the namespace in the inscope namespace map
    prefix = None
    if elem.namespace_name:
        inscope = {}
        for namespace in elem.inscope_namespaces:
            inscope[namespace.namespace_name] = namespace.prefix
        prefix = inscope[elem.namespace_name]
        if prefix:
            prefix += ':'
    if not prefix:
        prefix = ''

    if len(list(elem.attributes)) + len(list(elem.namespace_attributes)) == 0:
        # Write complete start tag (without attributes)
        f.write("\t"*depth + "<"+prefix+elem.local_name+'>\n')
    else:
        # Write start tag without the closing '>'
        f.write("\t"*depth + "<"+prefix+elem.local_name+'\n')

        # Write namespace attributes on separate lines
        for attr in elem.namespace_attributes:
        writeNSAttribute(f,attr,depth+1)
```

```
            # Write attributes on separate lines
            for attr in elem.attributes:
            writeAttribute(f,attr,depth+1)
            # Close the start tag
            f.write("\t"*depth + ">\n")

        # Write all element's children
        writeChildren(f,elem,depth+1)

        # Write end tag
        f.write("\t"*depth + "</"+prefix+elem.local_name+">\n")


def writeInstance(instance,filename):
    """Ouptput XML for the given instance where each element is indented by tabs
    and each attribute is placed on a separate line"""

    # Open output file
    f = open(filename,'w')
    # Write the content of the XML instance document
    writeChildren(f,instance.document,0)
    # Close output file
    f.close()


def on_xsi_valid(job,instance):
    """This method will be automatically called by RaptorXML after successful
    validation of the XML instance"""

    # Create a 'output.xml' file in the job's ouptut directory (when run from the
    CLI this will be the current working directory)
    filename = os.path.join(job.output_dir,'output.xml')
    # Write a reformatted version of the instance XML file where each attribute
    is placed on a separate line
    writeInstance(instance,filename)
    # Register the newly generated 'output.xml' output file
    job.append_output_filename(filename)
```

 © 2014 Altova GmbH

5.4.2 Ergebnisdokument

Im Folgenden finden Sie den vom Python-Skript reformat.py erzeugten Code des Dokuments output.xml.

Dateiname: output.xml

```
<OrgChart
	@xmlns="http://www.xmlspy.com/schemas/orgchart"
	@xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	@xmlns:ipo2="http://www.altova.com/IPO"
	@xmlns:ts="http://www.xmlspy.com/schemas/textstate"
	@xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
>
	<CompanyLogo
		@href="http://www.altova.com/nanonull.gif"
	>
	</CompanyLogo>
	<Name>
		Organization Chart
	</Name>
	<Office>
		<Name>
			Nanonull, Inc.
		</Name>
		<Desc>
			<para>
				The company was established
				<Style
					@css="font-weight: bold"
				>
					in Beverly in 1995
				</Style>
				 as a privately held software company. Since 1996,
Nanonull has been actively involved in developing nanoelectronic software
technologies. It released the first version of its acclaimed
				<Style
					@css="font-style: italic"
				>
					NanoSoft Development Suite
				</Style>
				 in February 1999. Also in 1999, Nanonull increased
its capital base with investment from a consortium of private investment firms.
The company has been expanding rapidly ever since.
			</para>
			<para>
				Due to the fact that nanoelectronic software
components are new and that sales are restricted to corporate customers,
Nanonull and its product line have not received much media publicity in the
company's early years. This has however changed in recent months as trade
journals have realized the importance of this revolutionary technology.
```

```
        </para>
</Desc>
<Location>
        US
</Location>
<Address
        @xsi:type="ipo2:US-Address"
>
        <ipo2:street
                @xmlns:ipo="http://www.altova.com/IPO"
        >
                900 Cummings Center
        </ipo2:street>
        <ipo2:city>
                Boston
        </ipo2:city>
        <ipo2:state>
                MA
        </ipo2:state>
        <ipo2:zip>
                3234
        </ipo2:zip>
</Address>
<Phone>
        +1 (321) 555 5155 0
</Phone>
<Fax>
        +1 (321) 555 5155 4
</Fax>
<EMail>
        office@nanonull.com
</EMail>
<Department>
        <Name>
                Administration
        </Name>
        <Person
                @union="fred"
        >
                <First>
                        Vernon
                </First>
                <Last>
                        Callaby
                </Last>
                <Title>
                        Office Manager
                </Title>
                <PhoneExt>
                        582
                </PhoneExt>
                <EMail>
                        v.callaby@nanonull.com
```

 © 2014 Altova GmbH

```
                        </EMail>
                        <Shares>
                               1500
                        </Shares>
                        <LeaveTotal>
                               25
                        </LeaveTotal>
                        <LeaveUsed>
                               4
                        </LeaveUsed>
                        <LeaveLeft>
                               21
                        </LeaveLeft>
                        <union>
                               3
                        </union>
                        <list>
                               abc def
                        </list>
                        <bool>
                               true
                        </bool>
                        <idref>
                               fred
                        </idref>
                        <idrefs>
                               fred joe
                        </idrefs>
                        <entity>
                               myUnparsedEntity
                        </entity>
                        <notation>
                               Altova-Orgchart
                        </notation>
                 </Person>
                 ...
          </Department>
          ...
    </Office>
    ...
</OrgChart>
```

5.5 Python-API: Das Auftragsobjekt

Klasse **Job**
Eine Job-Klasse repräsentiert einen Validierungsauftrag in RaptorXML.

Die Klasse Job stellt das folgende (schreibgeschützte) **Instanzattribut** bereit:

Job.output_dir
Gibt (bei Ausführung im Server-Modus) eine Datei-URL mit dem Ausgabeverzeichnis für den Auftrag zurück, andernfalls wird das aktuelle Arbeitsverzeichnis zurückgegeben.

Job.script_params
Gibt ein dict (vordefinierte Python-Datenstruktur) mit den benutzerdefinierten Script-Parametern zurück (bei Verwendung der Option --script-param).

Die Klasse Job stellt die folgende **Instanzmethode** bereit:

Job.append_ouptput_filename(filename)
Fügt einen zusätzlichen Ausgabedateinamen zur Liste der Ausgabedateien für den Auftrag hinzu. Diese Liste wird in der Ausgabe der Befehlszeile und im über die HTTP-Schnittstelle zurückgegebenen Ergebnisdokument angezeigt.

© 2014 Altova GmbH

5.6 Python-XML-API

Das **xml**-Modul bietet eine Python-Schnittstelle für die XML Infoset-Spezifikation. Dabei wird die zugrunde liegende C++ Infoset-Implementierung verwendet. Über diese Python-Schnittstelle kann der Benutzer durch die XML-Dokumentstruktur navigieren und Informationen aus jedem beliebigen XML-Node aufrufen.

Verfügbare Typen
Es stehen die folgenden Typen zur Verfügung. Eine genauere Beschreibung dazu finden Sie in den Unterabschnitten dieses Abschnitts.

Klasse xml.Document
Die Klasse `Document` repräsentiert ein XML-Dokument und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Document Information Item verfügbar.

Klasse xml.Element
Die Klasse `Element` repräsentiert ein XML-Element und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Element Information Item verfügbar.

Klasse xml.Attribute
Die Klasse `Attribute` repräsentiert ein XML-Attribut und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Attribute Information Item verfügbar.

Klasse xml.NSAttribute
Die Klasse `NSAttribute` repräsentiert ein XML-Namespace-Attribut und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Attribute Information Item verfügbar.

Klasse xml.ProcessingInstruction
Die Klasse `ProcessingInstruction` repräsentiert eine XML-Verarbeitungsanweisung und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Processing Instruction Information Item verfügbar.

Klasse xml.UnexpandedEntityReference
Die Klasse `UnexpandedEntityReference` repräsentiert eine nicht erweiterte XML-Entity-Referenz und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Unexpanded Entity Reference Information Item verfügbar.

Klasse xml.Character
Die Klasse `Character` repräsentiert XML-Zeichendaten und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Character Information Item verfügbar.

Klasse xml.Comment
Die Klasse `Comment` repräsentiert einen XML-Kommentar und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Comment Information Item verfügbar.

Klasse xml.UnparsedEntity
Die Klasse `UnparsedEntity` repräsentiert eine ungeparste XML-Entity und macht die

Eigenschaften des in der XML Infoset-Spezifikation definierten Unparsed Entity Information Item verfügbar.

Klasse xml.Notation

Die Klasse `Notation` repräsentiert eine XML-Notation und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Notation Information Item verfügbar.

Klasse xml.Namespace

Die Klasse `Namespace` repräsentiert einen XML-Namespace und macht die Eigenschaften des in der XML Infoset-Spezifikation definierten Namespace Information Item verfügbar.

Klasse xml.QName

Die Klasse `QName` repräsentiert einen qualifizierten XML-Namen.

© 2014 Altova GmbH

5.6.1 xml.Attribute

Klasse **xml.Attribute**
Ein `Attribute` repräsentiert ein XML Attribute Information Item (Informationseinheit für XML-Attribute). Es repräsentiert nur normale XML-Attribute, nicht spezielle Namespace-bindende XML-Attribute. Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

Attribute.namespace_name
Der Namespace-Name des Attributs, sofern einer existiert. Ansonsten ist dieses Attribut `None`.

Attribute.local_name
Der lokale Teil eines Attributnamens. Dies schließt kein Namespace-Präfix oder einen folgenden Doppelpunkt ein.

Attribute.prefix
Das Namespace-Präfix eines Attributnamens. Besitzt der Name kein Präfix, ist dieses Attribut `None`.

Attribute.normalized_value
Der normalisierte Attributwert.

Attribute.specified
Ein Flag, das anzeigt, ob dieses Attribut direkt im Start-Tag seines Elements angegeben wurde oder über eine Voreinstellung in der DTD.

Attribute.owner_element
Die Informationseinheit für Elemente, die diese Informationseinheit in ihrem `attributes` Attribut enthält.

5.6.2 xml.Character

Klasse **xml.Character**
Ein `Character` Objekt repräsentiert XML Character Information Items (Informationseinheit für Zeichendaten). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

Character.character_code
Ein String, der alle zusammengeführten nebeneinander liegenden Codepunkte der Informationseinheit für Zeichendaten enthält.

Character.element_content_whitespace
Ein Boolescher Wert, der anzeigt, ob das im Elementinhalt auftauchende Zeichen ein Whitespace-Zeichen.

Character.parent
Die Informationseinheit für Elemente, die diese Informationseinheit in ihrer `children`-Eigenschaft enthält.

 © 2014 Altova GmbH

5.6.3 xml.Comment

Klasse **xml.Comment**
Ein `Comment` Objekt repräsentiert ein XML Comment Information Item (Informationseinheit für Kommentare). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

Comment.content
Ein String, der den Inhalt des Kommentars repräsentiert.

Comment.parent
Die Informationseinheit für Dokumentelemente oder Elemente, die diese Informationseinheit in ihrer `children`-Eigenschaft enthält.

5.6.4 xml.Document

Klasse **`xml.Document`**
Ein `Document` Objekt repräsentiert ein XML Document Information Item (Informationseinheit für XML-Dokumentelemente). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

`Document.children`
Eine geordnete Liste von untergeordneten Informationseinheiten (Information Items) in Dokumentreihenfolge. Die Liste enthält genau eine Informationseinheit für Elemente. Die Liste beinhaltet ebenfalls eine Informationseinheit für Verarbeitungsanweisungen für jede Verarbeitungsanweisung außerhalb des Dokumentelements und eine Informationseinheit für Kommentare für jeden Kommentar außerhalb des Dokumentelements. Verarbeitungsanweisungen und Kommentare innerhalb der DTD sind davon ausgeschlossen. Gibt es eine Dokumenttyp-Deklaration, enthält die Liste ebenfalls eine Informationseinheit für Dokumenttyp-Deklarationen.

`Document.document_element`
Die Informationseinheit für Elemente, die zum Dokumentelement gehört.

`Document.notations`
Eine unsortierte Gruppe mit Informationseinheiten für Notationen, eine für jede Notation, die in der DTD deklariert ist.

`Document.unparsed_entities`
Eine unsortierte Gruppe mit Informationseinheiten für ungeparste Entities, eine für jede ungeparste Entity, welche in der DTD deklariert ist.

`Document.base_URI`
Die Basis-URI der Dokument-Entity.

`Document.character_encoding_scheme`
Der Name des Zeichenverschlüsselungsschemas, in dem die Dokument-Entity ausgedrückt ist.

`Document.standalone`
Die Anzeige des Standalone-Status eines Dokuments, entweder `True` oder `False`. Dieses Attribut geht aus der optionalen Standalone-Dokumentdeklaration in der XML-Deklaration am Anfang der Dokument-Entity hervor und gibt `None` zurück, wenn es keine Standalone-Dokumentdeklaration gibt.

`Document.version`
Ein String, welcher die XML-Version des Dokuments repräsentiert. Diese Eigenschaft wird aus der optionalen XML-Deklaration am Anfang einer Dokument-Entity abgeleitet und gibt `None` zurück, wenn es keine XML-Deklaration gibt.

 © 2014 Altova GmbH

5.6.5 xml.Namespace

Klasse **xml.Namespace(prefix, namespace_name)**
Ein `Namespace` Objekt repräsentiert ein XML-Namespace Binding und erzeugt ein Objekt der Klasse `Namespace`. Alle Argumente sind obligatorisch. Die Klasse `Namespace` stellt die folgenden Instanzattribute zur Verfügung:

Namespace.prefix
Das Präfix, dessen Bindung diese Informationseinheit beschreibt. Syntaktisch ist das der Teil des Attributnamens, der dem Präfix `xmlns:` folgt. Ist der Attributname einfach `xmlns`, so dass die Deklaration dem Standard-Namespace entspricht, ist dieses Attribut `None`.

Namespace.namespace_name
Der Namespace-Name, an den das Präfix gebunden ist.

5.6.6 xml.Element

Klasse **xml.Element**
Ein `Element` Objekt repräsentiert ein XML Element Information Item (Informationseinheit für XML-Elemente). Es stellt die folgenden (schreibgeschützten) Instanzattribute bereit:

Element.namespace_name
Der Namespace-Name des Elementtyps, sofern einer existiert. Gehört das Element nicht zu einem Namespace, ist diesem Attribut kein Wert zugewiesen.

Element.local_name
Der lokale Teil des Elementtypnamens. Dies schließt kein Namespace-Präfix oder einen folgenden Doppelpunkt ein.

Element.prefix
Das Namespace-Präfix des Elementtypnamens. Besitzt der Name kein Präfix, ist dieses Attribut `None`.

Element.children
Eine geordnete Liste mit Informationseinheiten für untergeordnete Informationseinheiten in Dokumentreihenfolge. Diese Liste enthält Informationseinheiten für Elemente, Verarbeitungsanweisungen, nicht erweiterte Entity-Referenzen, Zeichendaten und Kommentare, eine Einheit für jedes Element, für jede Verarbeitungsanweisung, für jede Referenz auf eine nicht verarbeitete externe Entity, für alle Zeichendaten und für jeden Kommentar, der direkt im aktuellen Element erscheint. Ist das Element leer, hat diese Liste keinen Inhalt.

Element.attributes
Eine ungeordnete Zusammenstellung mit Informationseinheiten für Attribute, eine Einheit für jedes Attribut (angegeben oder als Voreinstellung aus der DTD) dieses Elements. Namespace-Deklarationen erscheinen nicht in dieser Zusammenstellung. Hat das Element keine Attribute, hat diese Liste keinen Inhalt.

Element.namespace_attributes
Eine ungeordnete Zusammenstellung mit Informationseinheiten für Attribute, eine für jede Namespace-Deklaration (angegeben oder als Voreinstellung aus der DTD) dieses Elements. Eine Deklaration der Form `xmlns=""` und `xmlns:name=""` die den Standard-Namespace bzw. das Standard-Präfix aufhebt, zählt als Namespace-Deklaration. Die Aufheben von Präfixen wurde in Namespaces in XML 1.1 hinzugefügt. Per Definition haben alle Namespace-Attribute (eingeschlossen die mit xmlns benannten, deren `prefix`-Attribut keinen Wert hat) die Namespace-URI `http://www.w3.org/2000/xmlns/`. Hat das Element keine Namespace-Deklarationen, hat diese Zusammenstellung keinen Inhalt.

Element.inscope_namespaces
Eine ungeordnete Zusammenstellung mit Informationseinheiten für Namespaces, eine für jeden Namespace, der für dieses Element gültig ist. Diese Zusammenstellung enthält immer eine Einheit mit dem Präfix xml, das implizit an den Namespace-Namen `http://www.w3.org/XML/1998/namespace` gebunden ist. Es enthält keine Einheit mit dem Präfix xmlns (welches zur Namespace-Deklaration verwendet wird), weil eine Anwendung niemals auf ein Element oder Attribut mit diesem Präfix stoßen kann. Die Zusammenstellung wird alle Informationseinheiten für Namespaces enthalten, die auch in der Liste der `namespace_attributes` zu finden sind, mit Ausnahme derer, die eine Deklaration der Form

 © 2014 Altova GmbH

`xmlns=""` oder `xmlns:name=""` repräsentieren und somit keinen Namespace deklarieren, sondern den Standard-Namespace und das Standard-Präfix aufheben. Wenn die Präfixe der qualifizierten Namen aufgelöst werden, sollte dieses Attribut vorrangig zum Attribut `namespace_attributes` verwendet werden.

`Element.base_URI`
Die Basis-URI des Elements.

`Element.parent`
Die Informationseinheit für Dokumentelemente oder Elemente, die diese Informationseinheit in ihrem Attribut `children` enthält.

5.6.7 xml.Notation

Klasse **xml.Notation**
Ein `Notation` Objekt repräsentiert ein XML Notation Information Item (Informationseinheit für Notationen). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

Notation.name
Der Name der Notation.

Notation.system_identifier
Der System Identifier der Notation, wie er in der Deklaration der Notation erscheint. Wurde kein System Identifier angegeben, ist dieses Attribut `None`.

Notation.public_identifier
Der Public Identifier der Notation. Hat die Notation keinen Public Identifier, ist dieses Attribut `None`.

Notation.declaration_base_URI
Die Basis-URI, relativ zu der der System Identifier aufgelöst werden soll.

 © 2014 Altova GmbH

5.6.8 xml.NSAttribute

Klasse **xml.NSAttribute**
Ein `NSAttribute` repräsentiert ein XML Attribute Information Item (Informationseinheit für Attribute)speziell für das Namespace Binding. Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

NSAttribute.namespace_name
Der Namespace-Name, der immer http://www.w3.org/2000/xmlns/ ist.

NSAttribute.local_name
Der lokale Teil eines Attributnamens. Dies schließt kein Namespace-Präfix oder einen folgenden Doppelpunkt ein.

NSAttribute.prefix
Das Namespace-Präfix eines Attributnamens. Besitzt der Name kein Präfix, ist dieses Attribut `None`.

NSAttribute.normalized_value
Der normalisierte Attributwert.

NSAttribute.specified
Ein Flag, das anzeigt, ob dieses Attribut direkt im Start-Tag seines Elements angegeben wurde oder über eine Voreinstellung in der DTD.

NSAttribute.owner_element
Die Informationseinheit für Elemente, die diese Informationseinheit in ihrem `attributes` Attribut enthält.

5.6.9 xml.ProcessingInstruction

Klasse **`xml.ProcessingInstruction`**
Ein `ProcessingInstruction` Objekt repräsentiert ein XML Processing Instruction Information Item (Informationseinheit für XML-Verarbeitungsanweisungen). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

`ProcessingInstruction.target`
Ein String, der das Ziel der Verarbeitungsanweisung repräsentiert.

`ProcessingInstruction.content`
Ein String, der den Inhalt der Verarbeitungsanweisung repräsentiert, ausgeschlossen das Ziel und alle Leerzeichen, die direkt darauf folgen. Gibt es einen solchen Inhalt nicht, ist der Wert dieser Eigenschaft ein leerer String.

`ProcessingInstruction.parent`
Die Informationseinheit für Dokumentelemente, Elemente oder Dokumenttyp-Deklarartionen, die diese Informationseinheit in ihrem `children`-Attribut enthält.

 © 2014 Altova GmbH

5.6.10 xml.QName

Klasse **xml.QName(local_name, namespace_name)**
Ein QName Objekt repräsentiert einen qualifizierten XML-Namen und erzeugt ein Objekt der Klasse QName. Alle Argumente sind obligatorisch. Die Klasse QName stellt die folgenden Instanzattribute zur Verfügung:

QName.local_name
Der lokale Teil des qualifizierten Namens.

QName.namespace_name
Der Namespace-Teil des qualifizierten Namens.

5.6.11 xml.UnexpandedEntityReference

Klasse **xml.UnexpandedEntityReference**
Ein `UnexpandedEntityReference` Objekt repräsentiert ein Unexpanded XML Entity Reference Information Item (Informationseinheit für nicht erweiterte Entity-Verweise). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

UnexpandedEntityReference.name
Der Name der referenzierten Entity.

UnexpandedEntityReference.parent
Die Informationseinheit für Elemente, die diese Informationseinheit in ihrem `children` Attribut enthält.

 © 2014 Altova GmbH

5.6.12 xml.UnparsedEntity

Klasse **xml.UnparsedEntity**
Ein `UnparsedEntity` Objekt repräsentiert ein Unparsed XML Entity Information Item (Informationseinheit für ungeparste XML-Entities). Es stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

UnparsedEntity.name
Der Name der Entity.

UnparsedEntity.system_identifier
Der System Identifier der Entity, wir er in der Deklaration der Entity erscheint.

UnparsedEntity.public_identifier
Der Public Identifier der Entity. Wenn die Entity keinen Public Identifier hat, ist dieses Attribut `None`.

UnparsedEntity.declaration_base_URI
Die Basis-URI, zu der der System Identifier aufgelöst werden soll.

UnparsedEntity.notation_name
Der Notationsname, der mit der Entity verbunden ist.

UnparsedEntity.notation
Die Informationseinheit für Notationen, benannt nach dem Notationsnamen. Gibt es keine Deklaration für eine Notation mit diesem Namen, ist dieses Attribut `None`.

5.7 Python-XSD-API

Das **xsd**-Modul bietet eine Python-Schnittstelle für die C++-Implementierung für die XML-Schema-Datenmodellebene. Über diese Python-Schnittstelle kann der Benutzer durch die XML-Schemadokumentstruktur und das Post Schema Validation Infoset (PSVI) navigieren und Informationen daraus aufrufen.

Verfügbare Typen
Es sehen die folgenden Typen zur Verfügung. Eine nähere Beschreibung dazu finden Sie in den Unterabschnitten dieses Abschnitts.

Klasse xsd.Annotation
Die Klasse `Annotation` repräsentiert menschen- und maschinenlesbare Annotationen von Schemakomponenten.

Klasse xsd.Any
Eine `Any`-Klasse dient zur Validierung von Informationseinheiten für Attribute und Elemente, abhängig vom Namespace-Namen und optional auch abhängig von ihren lokalen Namen.

Klasse xsd.AnyAttribute
Eine `AnyAttribute`-Klasse dient zur Validierung von Informationseinheiten für Attribute, abhängig vom Namespace-Namen und optional auch abhängig von ihren lokalen Namen.

Klasse xsd.Assertion
Die Klasse `Assertion` schränkt das Vorhandensein und die Werte miteinander in Zusammenhang stehender Elemente und Attribute ein.

Klasse xsd.AttributeDeclaration
Eine `AttributeDeclaration` Klasse dient zu folgenden Zwecken: (i) lokale Validierung der Werte von Informationseinheiten für Attribute anhand einer simpleType-Definition und (ii) Definieren von Standardwerten oder festgelegten Werten für Informationseinheiten für Attribute.

Klasse xsd.AttributeGroupDefinition
Eine `AttributeGroupDefinition`-Klasse wird nicht zur eigentlichen Validierung verwendet, erzeugt aber zur Gänze oder teilweise eine oder mehrere complexType-Definitionen. Attributgruppen werden nach ihrem `name` und `target namespace` identifiziert und müssen innerhalb eines XSD-Schemas eindeutig sein.

Klasse xsd.AttributePSVI
Die Klasse `AttributePSVI` enthält PSVI-Informationen zu einem Attribut.

Klasse xsd.AttributeUse
Die Klasse `AttributeUse` repräsentiert menschen- und maschinenlesbare Annotationen von Schemakomponenten.

Klasse xsd.Block
Die Klasse `Block` bildet Teil der Definition einer Elementdeklaration im Schema.

Klasse xsd.ComplexTypeDefinition

 © 2014 Altova GmbH

Eine `ComplexTypeDefinition`-Klasse definiert die Eigenschaften eines complexType mittels ihrer Instanzattribute.

Klasse xsd.ContentType
Eine `ContentType`-Klasse definiert den Inhaltstyp des Elements.

Klasse xsd.Defined
Die Klasse `Defined` repräsentiert ein Schlüsselwortmitglied der Wertegruppe, die für das `disallowed_names` Attribut von `NamespaceConstraint` zulässig ist.

Klasse xsd.DerivationMethod
Eine `DerivationMethod`-Klasse enthält Informationen über die Ableitungsmethode.

Klasse xsd.ENTITY
Die Klasse `ENTITY` repräsentiert den `ENTITY`-Attributtyp von XML.

Klasse xsd.ElementDeclaration
Die Klasse `ElementDeclaration` dient zu folgenden Zwecken: (i) lokale Validierung der Werte von Informationseinheiten für Elemente anhand einer Typdefinition; (ii) Definieren von Standardwerten oder festgelegten Werten für Informationseinheiten für Elemente; (iii) Definieren von Eindeutigkeit und von Beziehungen zur Referenzeinschränkung für die Werte miteinander in Zusammenhang stehender Elemente und Attribute; (iv) Festlegen der Ersetzbarkeit von Elemente über den Mechanismus von Elementersetzungsgruppen.

Klasse xsd.ElementPSVI
Wenn die Schemagültigkeit einer Informationseinheit für Elemente überprüft wurde, werden die PSVI-Eigenschaften in Instanzattributen der Klasse `ElementPSVI` zurückgegeben.

Klasse xsd.Final
Ein complexType mit einer leeren Definition für `Final` kann als Basistypdefinition für andere durch Erweiterung oder Einschränkung abgeleitete Typen verwendet werden; die expliziten Werte `extension` und `restriction` verhindern weitere Ableitungen durch Erweiterung bzw. Einschränkung. Wenn alle Werte definiert sind, gilt der complexType als final, da keine weiteren Ableitungen möglich sind.

Klasse xsd.ID
Die Klasse `ID` repräsentiert den `ID`-Attributtyp von XML.

Klasse xsd.IDREF
Die Klasse `IDREF` repräsentiert eine Sequenz von XML-`ID`-Attributtypen.

Klasse xsd.ID_IDREF_binding
Die Klasse `ID_IDREF_binding` repräsentiert ein Binding zwischen `ID` und `IDREF`.

Klasse xsd.ID_IDREF_table
Die Klasse `ID_IDREF_table` repräsentiert eine Gruppe von `ID-IDREF mappings`.

Klasse xsd.IdentityConstraintDefinition
Die Klasse `IdentityConstraintDefinition` dient zur Definition der Eindeutigkeit und enthält Referenzeinschränkungen hinsichtlich des Inhalts mehrerer Elemente und Attribute.

Klasse xsd.Instance
Die Klasse `Instance` repräsentiert das Instanzdokument.

Klasse xsd.ModelGroup
Die Klasse `ModelGroup` definiert eine sequenzielle (`sequence`), trennende (`choice`) oder verbindende (`all`) Interpretation ihres Attributs.

Klasse xsd.ModelGroupDefinition
Eine Klasse `ModelGroupDefinition` wird durch ihren `name` und `target namespace` identifiziert. Elementgruppen (Model Groups) müssen innerhalb eines XSD-Schemas eindeutig sein. Elementgruppendefinitionen werden bei der Validierung nicht berücksichtigt, doch kann der `term` eines `Particle` zur Gänze oder zum Teil einer Elementgruppe `ModelGroup` aus einer `ModelGroupDefinition` entsprechen. Das Instanzattribut `model_group` ist die Elementgruppe `ModelGroup`, für die `ModelGroupDefinition` einen Namen liefert.

Klasse xsd.NCName
Die Klasse `NCName` repräsentiert einen NCName (non-colonized name).

Klasse xsd.NMTOKEN
Die Klasse `NMTOKEN` repräsentiert den `NMTOKEN` Attributtyp aus XML.

Klasse xsd.NOTATION
Die Klasse `NOTATION` repräsentiert den `NOTATION`-Attributtyp aus XML.

Klasse xsd.Name
Die Klasse `Name` repräsentiert einen XML-Namen.

Klasse xsd.NamespaceBinding
Die Klasse `NamespaceBinding` ermöglicht die Bindung eines `namespace` an ein `prefix`.

Klasse xsd.NamespaceConstraint
Die Klasse `NamespaceConstraint` dient zur Validierung von Informationseinheiten für Attribute und Elemente, die entsprechend dem angegebenen Constraint ausgewählt wurden.

Klasse xsd.NotationDeclaration
Eine Klasse `NotationDeclaration` definiert einen gültigen Element- oder Attributwert. Notationsdeklaration werden bei der Validierung nicht berücksichtigt. Sie werden allerdings beim Validieren von Strings als Mitglieder des simpleType `NOTATION` referenziert. Eine Informationseinheit für Attribute oder Elemente, deren verwendete Typdefinition oder Validierungstyp vom simpleType `NOTATION` abgeleitet ist, ist nur dann gültig, wenn ihr Wert sich in den Enumerationen eines solchen simpleType befindet. Folglich muss ein solcher Wert der `name` einer Notationsdeklaration sein.

Klasse xsd.OpenContent
Ein `OpenContent` Eigentschaftseintrag. Optional, wenn `variety` "element-only" oder "mixed" ist, andernfalls darf er nicht vorhanden sein.

Klasse xsd.PSVI
Die Klasse `PSVI` dient zur Prüfung der Schemagültigkeit von Elementen und Attributen.

Klasse xsd.Particle
Eine `Particle`-Klasse enthält die Komponenten. Diese Komponenten sind entweder direkt oder indirekt enthalten. Direkt enthalten ist die Komponente, die der Wert ihres `term` Attributs ist. Indirekt enthalten sind die Partikel, Gruppen, Wildcards und Elementdeklarationen, die im Wert ihrer `term` Eigenschaft enthalten sind.

 © 2014 Altova GmbH

Klasse xsd.QName
Die Klasse `QName` repräsentiert einen qualifizierten XML-Namen.

Klasse xsd.Schema
Die Klasse `schema` enthält eine Sammlung von Schemakomponenten, z.B. Typdefinitionen und Elementdeklarationen, die einen gemeinsamen Ziel-Namespace haben.

Klasse xsd.Scope
Die Klasse `Scope` repräsentiert einen `Scope`-Eigenschaftseintrag. Obligatorisch.

Klasse xsd.Sibling
Die Klasse `Sibling` repräsentiert ein Keyword Member der für das `disallowed_names` Attribut von `NamespaceConstraint` zulässigen Wertegruppe.

Klasse xsd.SimpleTypeDefinition
Die Klasse `SimpleTypeDefinition` repräsentiert durch ihr `name`- und `target namespace`-Attribut identifizierte simpleTypes.

Klasse xsd.TypeAlternative
Die Klasse `TypeAlternative` wird von einer `ElementDeclaration` zur Definition einer Bedingung (`test`) verwendet, unter der ein bestimmter Typ (`type_definition`) als geltende Typdefinition für Informationseinheiten für Elemente, für die diese `ElementDeclaration` gilt, verwendet wird. Jede `ElementDeclaration` kann in ihrer Typtabelle (`TypeTable`) mehrere Typalternativen haben.

Klasse xsd.TypeTable
Die Typdefinition, anhand welcher eine Informationseinheit für Elemente validiert wird (die geltende Typdefinition) kann eine andere sein, als die deklarierte Typdefinition. Aufgrund der `TypeTable`-Eigenschaft einer `ElementDeclaration`, die die Zuweisung von Typen auf Basis von Bedingungen regelt, und aufgrund des `xsi:type`-Attributs einer Informationseinheit für Elemente können die geltende Typdefinition und die deklarierte Typdefinition unterschiedlich sein.

Klasse xsd.Unbounded
Die Klasse `Unbounded` ist ein String-Wert. Sie repräsentiert den oberen Wert der Eigenschaft `maxOccurs`.

Klasse xsd.ValueConstraint
Die Klasse `ValueConstraint` repräsentiert eine Eigenschaft der Klasse `AttributeUse`.

Klasse xsd.XPathExpression
Um eine Assertion zu überprüfen, wird eine Instanz des XPath 2.0-Datenmodells erzeugt, in der die überprüfte Informationseinheit für Elemente der Root Node (der oberste Node) ist und den Elementen und Attributen Typen und Werte gemäß den Regeln des XPath 2.0-Datenmodells zugewiesen werden. Bei Auswertung anhand dieser Datenmodelltestinstanz ergibt die Auswertung von `test` entweder `True` oder `False`.

Spezielle vordefinierte Datentypobjekte
`anyAtomicType`
`anySimpleType`
`anyURI`

Objekte vom Datentyp String

```
language
normalizedString
string
token
```

Objekte vom Datentyp Boolean

```
boolean
```

Objekte vom Datentyp Number

```
byte
decimal
double
float
int
integer
long
negativeInteger
nonNegativeInteger
nonPositiveInteger
positiveInteger
short
unsignedByte
unsignedInt
unsignedLong
unsignedShort
```

Objekte vom Datentyp Duration

```
dayTimeDuration
duration
yearMonthDuration
```

Objekte vom Datentyp Time und Number

```
date
dateTime
dateTimeStamp
gDay
gMonth
gYear
gYearMonth
time
```

Objekte vom Datentyp Binary

```
base64Binary
hexBinary
```

Facet-Objekte

```
assertionsFacet
enumerationFacet
fractionDigitsFacet
lengthFacet
maxExclusiveFacet
maxInclusiveFacet
maxLengthFacet
minExclusiveFacet
```

 © 2014 Altova GmbH

```
minInclusiveFacet
minLengthFacet
pattern
totalDigitsFacet
```

5.7.1 xsd.Annotation

Klasse **xsd.Annotation**
Die Klasse `Annotation` repräsentiert menschen- und maschinenlesbare Annotationen von Schemakomponenten. *Zur Beschreibung*.

Die Klasse `Annotation` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung.

`Annotation.application_information`
Eine Sequenz von Informationseinheiten für Elemente für die automatische Verarbeitung.

`Annotation.user_information`
Eine Sequenz von Informationseinheiten für Elemente, die für den Gebrauch durch Personen bestimmt ist.

© 2014 Altova GmbH

5.7.2 xsd.Any

Klasse **xsd.Any**
Eine `Any`-Klasse dient zur Validierung von Informationseinheiten für Attribute und Elemente, abhängig vom Namespace-Namen und optional auch abhängig von ihren lokalen Namen. *Zur Beschreibung*.

Die `Any`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

Any.SKIP
Keinerlei Einschränkungen: bei der Informationseinheit muss es sich lediglich um wohlgeformten XML-Code handeln.

Any.STRICT
Für die Informationseinheit muss eine Deklaration auf oberster Ebene vorhanden sein oder die Informationseinheit muss den Typ `xsi:type` haben und muss gemäß ihrem Typ gültig sein.

Any.LAX
Wenn für die Informationseinheit eine eindeutige Deklaration verfügbar ist, muss sie gemäß dieser Deklaration gültig sein. D.h.: Falls möglich, validieren, andernfalls ist keine Validierung erforderlich.

Die Klasse `Any` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung):

Any.annotations
Eine Sequenz von `Annotation`-Komponenten.

Any.namespace_constraint
Ein Namespace Constraint-Eigenschaftseintrag. Obligatorisch.

Any.process_contents
Steuert die Auswirkung bei Prüfung der durch Wildcards zulässigen Informationseinheiten. Erhält entweder `SKIP`, `STRICT` oder `LAX`. Obligatorisch.

5.7.3 xsd.AnyAttribute

Klasse **xsd.AnyAttribute**
Eine `AnyAttribute`-Klasse dient zur Validierung von Informationseinheiten für Attribute, abhängig vom Namespace-Namen und optional auch abhängig von ihren lokalen Namen. *Zur Beschreibung.*

Die Klasse `AnyAttribute` stellt die folgenden **Konstanten** zur Verfügung:

Any.SKIP
Keinerlei Einschränkungen: bei der Informationseinheit muss es sich lediglich um korrekten XML-Code handeln.

Any.STRICT
Für die Informationseinheit muss eine Deklaration auf oberster Ebene vorhanden sein oder die Informationseinheit muss den Typ `xsi:type` haben und muss gemäß ihrem Typ gültig sein.

Any.LAX
Wenn für die Informationseinheit eine eindeutige Deklaration verfügbar ist, muss sie gemäß dieser Deklaration gültig sein. D.h.: Falls möglich, validieren, andernfalls ist keine Validierung erforderlich.

Die Klasse `AnyAttribute` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung):

AnyAttribute.annotations
Eine Sequenz von `Annotation`-Komponenten.

AnyAttribute.namespace_constraint
Ein Namespace Constraint-Eigenschaftseintrag. Obligatorisch.

AnyAttribute.process_contents
Steuert die Auswirkung bei Prüfung der durch Wildcards zulässigen Informationseinheiten. Erhält entweder `SKIP`, `STRICT` oder `LAX`. Obligatorisch.

 © 2014 Altova GmbH

5.7.4 xsd.Assertion

Klasse **`xsd.Assertion`**
Eine `Assertion`-Klasse schränkt das Vorhandensein und die Werte miteinander in Zusammenhang stehender Elemente und Attribute ein. *Zur Beschreibung*.

Die Klasse `Assertion` stellt die folgenden **Instanzattribute** zur Verfügung:

`Assertion.annotations`
Eine Sequenz von `Annotation`-Komponenten.

`Assertion.test`
Ein XPath-Ausdruck. Obligatorisch.

5.7.5 xsd.AttributeDeclaration

Klasse **xsd.AttributeDeclaration**
Eine `AttributeDeclaration` Klasse dient zu folgenden Zwecken: (i) lokale Validierung der Werte von Informationseinheiten für Attribute anhand einer simpleType-Definition und (ii) Definieren von Standardwerten oder festgelegten Werten für Informationseinheiten für Attribute. *Zur Beschreibung*.

Die `AttributeDeclaration` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

AttributeDeclaration.annotations
Eine Sequenz von `Annotation`-Komponenten.

AttributeDeclaration.name
Ein `xs:NCName`-Wert. Obligatorisch.

AttributeDeclaration.target_namespace
Ein `xs:anyURI`-Wert. Optional.

AttributeDeclaration.type_definition
Eine simpleType-Definition-Komponente. Obligatorisch.

AttributeDeclaration.scope
Ein `Scope`-Eigenschaftseintrag. Obligatorisch.

AttributeDeclaration.value_constraint
Ein `Constraint`-Eigenschaftseintrag. Optional.

AttributeDeclaration.inheritable
Ein `xs:boolean`-Wert. Obligatorisch.

© 2014 Altova GmbH

5.7.6 xsd.AttributeGroupDefinition

Klasse **`xsd.AttributeGroupDefinition`**
Eine `AttributeGroupDefinition`-Klasse wird nicht zur eigentlichen Validierung verwendet, erzeugt aber zur Gänze oder teilweise eine oder mehrere complexType-Definitionen. Attributgruppen werden nach ihrem `name` und `target namespace` identifiziert und müssen innerhalb eines XSD-Schemas eindeutig sein. *Zur Beschreibung.*

Die Klasse `AttributeGroupDefinition` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

`AttributeGroupDefinition.annotations`
Eine Sequenz von `Annotation`-Komponenten.

`AttributeGroupDefinition.name`
Ein `xs:NCName`-Wert. Obligatorisch.

`AttributeGroupDefinition.target_namespace`
Ein `xs:anyURI`-Wert. Optional.

`AttributeGroupDefinition.attribute_uses`
Eine Gruppe von `Attribute Use`-Komponenten.

`AttributeGroupDefinition.attribute_wildcard`
Eine Wildcard-Komponente. Optional.

5.7.7 xsd.AttributePSVI

Klasse **xsd.AttributePSVI**
Die Klasse `AttributePSVI` enthält PSVI-Informationen zu einem Attribut. *Zur Beschreibung*.

Die Klasse `AttributePSVI` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

AttributePSVI.validity
Der zutreffende Fall unter den folgenden: falls streng validiert und lokal gültig, dann `valid`; falls streng validiert und lokal ungültig, dann `invalid`; andernfalls `notKnown`.

AttributePSVI.validation_attempted
Der zutreffende Fall unter den folgenden: falls streng validiert, dann `full`; andernfalls `none`.

AttributePSVI.attribute_declaration
eine zur geltenden Typdefinition isomorphe Informationseinheit.

AttributePSVI.schema_normalized_value
Wenn der normalisierte Wert des Attributs gemäß der geltenden Typdefinition gültig ist, dann gilt der normalisierte Wert wie validiert, andernfalls fehlt er.

AttributePSVI.schema_actual_value
Wenn der Schema-normalisierte Wert nicht fehlt, dann der entsprechende tatsächliche Wert, andernfalls fehlt er.

AttributePSVI.type_definition
Eine zur geltenden Typdefinition isomorphe Informationseinheit.

AttributePSVI.type_definition_type
`simple`.

AttributePSVI.type_definition_namespace
Der Ziel-Namespace der Typdefinition.

AttributePSVI.type_definition_anonymous
`True`, wenn der `name` der Typdefinition nicht vorhanden ist, andernfalls `False`.

AttributePSVI.type_definition_name
Der `name` der Typdefinition, wenn der `name` nicht fehlt. Wenn die Eigenschaft `name` der Typdefinition fehlt, können Schema-Prozessoren einen Wert bereitstellen, der diese Typdefinition unter denen im selben Ziel-Namespace eindeutig identifiziert, müssen dies aber nicht.

AttributePSVI.member_type_definition
Eine zum validierenden Typ des tatsächlichen Schemawerts isomorphe Informationseinheit

AttributePSVI.member_type_definition_namespace
Der Ziel-Namespace des validierenden Typs.

AttributePSVI.member_type_definition_anonymous

 © 2014 Altova GmbH

`True` wenn der `name` des validierenden Typs fehlt, andernfalls `False`.

`AttributePSVI.member_type_definition_name`
Der `name` des validierenden Typs, falls vorhanden.

`AttributePSVI.member_type_definitions`
Eine Sequenz von simpleType-Definitionen derselben Länge wie der tatsächliche Schemawert, wobei jede dieser Informationseinheiten zum validierenden Typ der entsprechenden Informationseinheit im tatsächlichen Schemawert isomorph ist.

5.7.8 xsd.AttributeUse

Klasse **xsd.AttributeUse**
Eine `AttributeUse` Klasse ist eine Hilfs-Komponente, die das Vorkommen und das vorgegebene Verhalten von Attribut-Deklarationen kontrolliert. Sie spielt die gleiche Rolle für Attribut-Deklarationen in complexTypes wie Partikel für Element-Deklarationen. *Zur Beschreibung*.

Die Klasse `AttributeUse` stellt die folgenden (schreibgeschützten) Instanzattribute zur Verfügung:

AttributeUse.annotations
Eine Sequenz von `Annotation`-Komponenten.

AttributeUse.required
Ein `xs:boolean`-Wert. Obligatorisch.

AttributeUse.attribute_declaration
Eine `AttributeDeclaration`-Komponente. Obligatorisch.

AttributeUse.value_constraint
Ein `ValueConstraint` Eigenschaftseintrag. Optional.

AttributeUse.inheritable
Ein `xs:boolean`-Wert. Obligatorisch.

 © 2014 Altova GmbH

5.7.9 xsd.Block

Klasse **xsd.Block**
Die Klasse `Block` bildet Teil der Definition einer Elementdeklaration im Schema. Obligatorisch. *Zur Definition.*

Die Klasse `Block` stellt die folgenden **Konstanten** zur Verfügung:

Block.NONE
Block.EXTENSION
Block.RESTRICTION
Block.SUBSTITUTION

5.7.10 xsd.ComplexTypeDefnition

Klasse **xsd.ComplexTypeDefinition**
Eine `ComplexTypeDefinition`-Klasse definiert die Eigenschaften eines complexType mittels ihrer Instanzattribute (*siehe Liste unten*). *Zur Beschreibung*.

Die Klasse `ComplexTypeDefinition` stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ComplexTypeDefinition.annotations
Eine Sequenz von `Annotation`-Komponenten.

ComplexTypeDefinition.name
Ein `xs:NCName`-Wert. Optional.

ComplexTypeDefinition.target_namespace
Ein `xs:anyURI` Wert. Optional.

ComplexTypeDefinition.base_type_definition
Eine Typdefinition. Obligatorisch.

ComplexTypeDefinition.final
Eine Untergruppe von `{extension, restriction}`.

ComplexTypeDefinition.context
Obligatorisch, wenn das `name` Instanzattribut (*siehe oben*) fehlt. Darf andernfalls nicht vorhanden sein. Entweder eine `ElementDeclaration` oder eine `ComplexTypeDefinition`.

ComplexTypeDefinition.derivation_method
Eine aus der Menge `{extension, restriction}`. Obligatorisch.

ComplexTypeDefinition.abstract
Ein `xs:boolean` Wert. Obligatorisch.

ComplexTypeDefinition.attribute_uses
Eine Gruppe von `AttributeUse`-Komponenten.

ComplexTypeDefinition.attribute_wildcard
Eine Wildcard-Komponente. Optional.

ComplexTypeDefinition.content_type
Ein `ContentType` Eigenschaftseintrag. Obligatorisch.

ComplexTypeDefinition.prohibited_substitutions
Eine Untergruppe von `{extension, restriction}`.

ComplexTypeDefinition.assertions
Eine Sequenz von `Assertion`-Komponenten.

 © 2014 Altova GmbH

5.7.11 xsd.ContentType

Klasse **xsd.ContentType**
Eine `ContentType` Klasse definiert den Inhaltstyp des Elements. *Zur Beschreibung*

Die `ContentType`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

ContentType.EMPTY
ContentType.SIMPLE
ContentType.ELEMENT_ONLY
ContentType.MIXED

Die `ContentType` -Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ContentType.variety
Einer der Werte `{empty, simple, element-only, mixed}`. Obligatorisch.

ContentType.particle
Eine `Particle`-Komponente. Obligatorisch, wenn `{variety}` "element-only" oder "mixed" ist, darf andernfalls nicht vorhanden sein.

ContentType.open_content
Ein `OpenContent` Eigenschaftseintrag. Optional, wenn `{variety}` "element-only" oder "mixed" ist, darf andernfalls nicht vorhanden sein.

ContentType.simple_type_definition
Eine `SimpleTypeDefinition`-Komponente. Obligatorisch, wenn `{variety}` "simple" ist, darf andernfalls nicht vorhanden sein.

5.7.12 xsd.Defined

Klasse **xsd.Defined**
Die Klasse `Defined` repräsentiert ein Schlüsselwortmitglied der Wertegruppe, die für das `disallowed_names` Attribut von `NamespaceConstraint` zulässig ist. *Zur Beschreibung*.

Die `ContentType`-Klasse stellt die folgende **Instanzmethode** zur Verfügung:

Defined.__str__()

© 2014 Altova GmbH

5.7.13 xsd.DerivationMethod

Klasse **xsd.DerivationMethod**
Eine `DerivationMethod`-Klasse enthält Informationen über die Ableitungsmethode. *Zur Beschreibung*.

Die `DerivationMethod`-Klasse stellt die folgenden **Konstanten** zur Verfügung

DerivationMethod.NONE
DerivationMethod.RESTRICTION
DerivationMethod.EXTENSION
DerivationMethod.LIST
DerivationMethod.UNION

5.7.14 xsd.ENTITY

Klasse **xsd.ENTITY**
Die Klasse ENTITY repräsentiert den ENTITY-Attributtyp von XML. *Zur Beschreibung*.

Die ENTITY -Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ENTITY.value
Ein String, der den Wert der Entity bereitstellt.

© 2014 Altova GmbH

5.7.15 xsd.ElementDeclaration

Klasse **xsd.ElementDeclaration**
Die Klasse `ElementDeclaration` dient zu folgenden Zwecken: (i) lokale Validierung der Werte von Informationseinheiten für Elemente anhand einer Typdefinition; (ii) Definieren von Standardwerten oder festgelegten Werten für Informationseinheiten für Elemente; (iii) Definieren von Eindeutigkeit und von Beziehungen zur Referenzeinschränkung für die Werte miteinander in Zusammenhang stehender Elemente und Attribute; (iv) Festlegen der Ersetzbarkeit von Elemente über den Mechanismus von Elementersetzungsgruppen. *Zur Beschreibung*.

Die `ElementDeclaration`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ElementDeclaration.annotations
Eine Sequenz von `Annotation`-Komponenten.

ElementDeclaration.name
Ein `xs:NCName`-Wert. Obligatorisch.

ElementDeclaration.target_namespace
Ein `xs:anyURI`-Wert. Optional.

ElementDeclaration.type_definition
Eine Typdefinition. Erforderlich.

ElementDeclaration.type_table
Ein `TypeTable`-Eigenschaftseintrag. Optional.

ElementDeclaration.scope
Ein `Scope`-Eigenschaftseintrag. Obligatorisch.

ElementDeclaration.value_constraint
Ein `ValueConstraint`-Eigenschaftseintrag. Optional.

ElementDeclaration.nillable
Ein `xs:boolean`-Wert. Obligatorisch.

ElementDeclaration.identity_constraint_definitions
Eine Gruppe von `IdentityConstraintDefinition`-Komponenten.

ElementDeclaration.substitution_group_affiliations
Eine Gruppe von `ElementDeclaration`-Komponenten.

ElementDeclaration.substitution_group_exclusions
Eine Untergruppe von `{extension, restriction}`.

ElementDeclaration.disallowed_substitutions
Eine Untergruppe von `{substitution, extension, restriction}`.

ElementDeclaration.abstract

Ein `xs:boolean`-Wert. Obligatorisch.

© 2014 Altova GmbH

5.7.16 xsd.ElementPSVI

Klasse **xsd.ElementPSVI**
Wenn die Schemagültigkeit einer Informationseinheit für Elemente überprüft wurde, werden die PSVI-Eigenschaften in Instanzattributen der Klasse `ElementPSVI` zurückgegeben. *Zur Beschreibung*.

Die `ElementPSVI`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ElementPSVI.validity
Der zutreffende Fall unter den folgenden: `valid`, `invalid` oder `notKNown`. Nähere Informationen dazu finden Sie in der *Beschreibung*.

ElementPSVI.validation_attempted
Der zutreffende Fall unter den folgenden: `full`, `none` oder `partial`. Nähere Informationen dazu finden Sie in der *Beschreibung*.

ElementPSVI.element_declaration
Eine zur geltenden Deklaration selbst isomorphe Informationseinheit.

ElementPSVI.nil
Wert `True`, wenn Klausel 3.2.3 Element lokal gültig (Element) erfüllt wird, andernfalls `False`.

ElementPSVI.schema_normalized_value
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.schema_actual_value
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.type_definition
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.type_definition_type
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.type_definition_namespace
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.type_definition_anonymous

Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.type_definition_name
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.member_type_definition
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.member_type_definition_namespace
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.member_type_definition_anonymous
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.member_type_definition_name
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.member_type_definitions
Wenn eine für eine Informationseinheit für Elemente eine geltende Typdefinition bekannt ist, so ist dies im Post-Schema-Validation-Infoset der Wert der Informationseinheit. Siehe *Informationseinheiten für Elemente*.

ElementPSVI.inherited_attributes
Eine Liste von vererbten Informationseinheiten für Attribute. Die Beschreibung dazu finden Sie *hier*.

 © 2014 Altova GmbH

5.7.17 xsd.Final

Klasse **xsd.Final**
Ein complexType mit einer leeren Definition für `Final` kann als Basistypdefinition für andere durch Erweiterung oder Einschränkung abgeleitete Typen verwendet werden; die expliziten Werte `extension` und `restriction` verhindern weitere Ableitungen durch Erweiterung bzw. Einschränkung. Wenn alle Werte definiert sind, gilt der complexType als final, da keine weiteren Ableitungen möglich sind. *Zur Beschreibung*.

Die `Final`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

Final.NONE
Final.EXTENSION
Final.RESTRICTION
Final.LIST
Final.UNION

5.7.18 xsd.ID

Klasse **xsd.ID**
Repräsentiert den ID-Attributtyp von XML. *Zur Beschreibung*.

Die ID-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ID.value
Ein String, der den Wert der ID angibt.

© 2014 Altova GmbH

5.7.19 xsd.IDREF

Klasse **xsd.IDREF**
Die Klasse IDREF repräsentiert eine Sequenz von XML-ID-Attributtypen. *Zur Beschreibung*.

Die IDREF-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

IDREF.value
Eine Sequenz von ID-Werten.

5.7.20 xsd.ID_IDREF_binding

Klasse **xsd.ID_IDREF_binding**
Die Klasse `ID_IDREF_binding` repräsentiert ein Binding zwischen `ID` und `IDREF`. *Zur Beschreibung*.

Die `ID_IDREF_binding`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ID_IDREF_binding.id
ID_IDREF_binding.binding

© 2014 Altova GmbH

5.7.21 xsd.ID_IDREF_table

Klasse **xsd.ID_IDREF_table**
Die Klasse `ID_IDREF_table` repräsentiert eine Gruppe von `ID-IDREF mappings`. *Zur Beschreibung*.

Die `ID_IDREF_table`-Klasse stellt die folgenden (schreibgeschützten) *Instanzmethoden* zur Verfügung:

ID_IDREF_table.__len__()
ID_IDREF_table.__iter__()

5.7.22 xsd.IdentityConstraintDefinition

Klasse **xsd.IdentityConstraintDefinition**
Die Klasse `IdentityConstraintDefinition` dient zur Definition der Eindeutigkeit und enthält Referenzeinschränkungen hinsichtlich des Inhalts mehrerer Elemente und Attribute. *Zur Beschreibung*.

Die `IdentityConstraintDefinition`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

IdentityConstraintDefinition.KEY
Die identitätseinschränkende Definition sichert Eindeutigkeit wie bei `unique`. Weiters sichert die Konstante `key`, dass der gesamte gewählte Inhalt tatsächlich solche Tupel hat.

IdentityConstraintDefinition.KEYREF
Die identitätseinschränkende Definition sichert eine Entsprechung hinsichtlich des vom `selector` identifizierten Inhalts der Tupel, die sich aus der Auswertung des XPath-Ausdrucks (bzw. der XPath-Ausdrücke) des Felds mit denen des referenzierten Schlüssels ergeben.

IdentityConstraintDefinition.UNIQUE
Die identitätseinschränkende Definition sichert Eindeutigkeit hinsichtlich des vom `selector` identifizierten Inhalts der Tupel, die sich aus der Auswertung des XPath-Ausdrucks (bzw. der XPath-Ausdrücke) des Felds ergeben.

Die `IdentityConstraintDefinition`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

IdentityConstraintDefinition.annotations
Eine Sequenz von `Annotation`-Komponenten.

IdentityConstraintDefinition.name
Ein `xs:NCName`-Wert. Obligatorisch.

IdentityConstraintDefinition.target_namespace
Ein `xs:anyURI`-Wert. Optional.

IdentityConstraintDefinition.identity_constraint_category
Der zutreffende Fall unter den folgenden: `{key, keyref, unique}`. Obligatorisch.

IdentityConstraintDefinition.selector
Ein `XPathExpression` Eigenschaftseintrag. Obligatorisch.

IdentityConstraintDefinition.fields
Eine Sequenz von `XPathExpression`-Eigenschaftseinträgen.

IdentityConstraintDefinition.referenced_key
Eine `IdentityConstraintDefinition`-Komponente. Obligatorisch, wenn `identity_constraint_category` `keyref` ist, darf andernfalls (wenn `identity_constraint_category` `key` oder `unique` ist) nicht vorhanden sein. Wenn ein Wert

 © 2014 Altova GmbH

vorhanden ist, muss seine `identity_constraint_category` `key` oder `unique` sein.

5.7.23 xsd.Instance

Klasse **xsd.Instance**
Die Klasse `Instance` repräsentiert das Instanzdokument. *Zur Beschreibung*.

Die `Instance`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

Instance.filename
Instance.document
Instance.psvi
Instance.schema

© 2014 Altova GmbH

5.7.24 xsd.ModelGroup

Klasse **xsd.ModelGroup**
Die Klasse `ModelGroup` definiert eine sequenzielle (`sequence`), trennende (`choice`) oder verbindende (`all`) Interpretation des `particles`-Attributs. *Zur Beschreibung*.

Die `ModelGroup`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

ModelGroup.ALL
Legt fest, ob die von der Elementgruppe validierten Children der Informationseinheit für Elemente den angegebenen `particles` entsprechen müssen. Die Elemente können in jeder beliebigen Reihenfolge vorkommen.

ModelGroup.CHOICE
Legt fest, ob die von der Elementgruppe validierten Children der Informationseinheit für Elemente genau einem der angegebenen `particles` entsprechen müssen.

ModelGroup.SEQUENCE
Legt fest, ob die von der Elementgruppe validierten Children der Informationseinheit für Elemente den angegebenen `particles` genau in der gleichen Reihenfolge entsprechen müssen.

Die `ModelGroup`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ModelGroup.annotations
Eine Sequenz von `Annotation`-Komponenten.

ModelGroup.compositor
Eines der folgenden `{all, choice, sequence}`. Obligatorisch.

ModelGroup.particles
Eine Sequenz von `Particle`-Komponenten.

5.7.25 xsd.ModelGroupDefinition

KLasse **xsd.ModelGroupDefinition**
Eine Klasse `ModelGroupDefinition` wird durch ihren `name` und `target namespace` identifiziert. Elementgruppen (Model Groups) müssen innerhalb eines XSD-Schemas eindeutig sein. Elementgruppendefinitionen werden bei der Validierung nicht berücksichtigt, doch kann der `term` eines `Particle` zur Gänze oder zum Teil einer Elementgruppe `ModelGroup` aus einer `ModelGroupDefinition` entsprechen. Das Instanzattribut `model_group` ist die Elementgruppe `ModelGroup`, für die `ModelGroupDefinition` einen Namen liefert. *Zur Definition*.

Die `ModelGroupDefinition`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ModelGroupDefinition.annotations
Eine Sequenz von `Annotation`-Komponenten.

ModelGroupDefinition.name
Ein `xs:NCName`-Wert. Obligatorisch.

ModelGroupDefinition.target_namespace
Ein `xs:anyURI`-Wert. Optional.

ModelGroupDefinition.model_group
Eine `ModelGroup`-Komponente. Obligatorisch.

 © 2014 Altova GmbH

5.7.26 xsd.NCName

Klasse **xsd.NCName**
Die Klasse NCName repräsentiert einen NCName (non-colonized name). *Zur Beschreibung*.

Die NCName-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

NCName.value

5.7.27 xsd.NMTOKEN

Klasse **xsd.NMTOKEN**
Die Klasse NMTOKEN repräsentiert den NMTOKEN Attributtyp aus XML. *Zur Beschreibung*.

Die NMTOKEN-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

NMTOKEN.value

© 2014 Altova GmbH

5.7.28 xsd.NOTATION

Klasse **xsd.NOTATION**
Die Klasse NOTATION repräsentiert den NOTATION-Attributtyp aus XML. *Zur Beschreibung*.

Die NOTATION-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

NOTATION.namespace_name
NOTATION.local_part

5.7.29 xsd.Name

Klasse **xsd.Name**
Die Klasse `Name` repräsentiert einen XML-Namen. *Zur Beschreibung.*

Die `Name`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

Name.value

© 2014 Altova GmbH

5.7.30 xsd.NamespaceBinding

Klasse **xsd.NamespaceBinding**
Die Klasse `NamespaceBinding` ermöglicht die Bindung eines `namespace` an ein `prefix`. *Zur Beschreibung*.

Die `NamespaceBinding`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

NamespaceBinding.prefix
NamespaceBinding.namespace

5.7.31 xsd.NamespaceConstraint

Klasse **xsd.NamespaceConstraint**
Die Klasse `NamespaceConstraint` dient zur Validierung von Informationseinheiten für Attribute und Elemente, die entsprechend dem angegebenen Constraint ausgewählt wurden. *Zur Beschreibung*.

Die `NamespaceConstraint`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

NamespaceConstraint.ANY
NamespaceConstraint.ENUMERATION
NamespaceConstraint.NOT

Die `NamespaceConstraint`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

NamespaceConstraint.variety
Eines von `{any, enumeration, not}`. Obligatorisch.

NamespaceConstraint.namespaces
Eine Gruppe, bei deren Mitgliedern es sich entweder um einen `xs:anyURI` Wert oder den Wert `absent` handelt. Obligatorisch.

NamespaceConstraint.disallowed_names
Eine Gruppe, bei deren Mitgliedern es sich entweder um einen `xs:QName` Wert oder eines der Schlüsselwörter `defined` oder `sibling` handelt. Obligatorisch.

 © 2014 Altova GmbH

5.7.32 xsd.NotationDeclaration

Klasse **xsd.NotationDeclaration**
Eine Klasse `NotationDeclaration` definiert einen gültigen Element- oder Attributwert. Notationsdeklaration werden bei der Validierung nicht berücksichtigt. Sie werden allerdings beim Validieren von Strings als Mitglieder des simpleType `NOTATION` referenziert. Eine Informationseinheit für Attribute oder Elemente, deren verwendete Typdefinition oder Validierungstyp vom simpleType `NOTATION` abgeleitet ist, ist nur dann gültig, wenn ihr Wert sich in den Enumerationen eines solchen simpleType befindet. Folglich muss ein solcher Wert der `name` einer Notationsdeklaration sein. *Zur Beschreibung*.

Die `NotationDeclaration`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

`NotationDeclaration.annotations`
Eine Sequenz von `Annotation`-Komponenten.

`NotationDeclaration.name`
Ein `xs:NCName`-Wert. Obligatorisch.

`NotationDeclaration.target_namespace`
Ein `xs:anyURI`-Wert. Optional.

`NotationDeclaration.system_identifier`
Ein `xs:anyURI`-Wert. Obligatorisch, wenn `public_identifier` fehlt, andernfalls optional.

`NotationDeclaration.public_identifier`
Ein public ID-Wert. Obligatorisch, wenn `system_identifier` fehlt, andernfalls optional.

5.7.33 xsd.OpenContent

Klasse **xsd.OpenContent**
Ein `OpenContent` Eigenschaftseintrag. Optional, wenn `variety` "element-only" oder "mixed" ist, andernfalls darf er nicht vorhanden sein. *Zur Beschreibung*.

Die `OpenContent`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

OpenContent.INTERLEAVE
OpenContent.SUFFIX

Die `OpenContent`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

OpenContent.mode
Eines von `{interleave, suffix}`. Obligatorisch.

OpenContent.wildcard
Eine Wildcard-Komponente. Obligatorisch.

© 2014 Altova GmbH

5.7.34 xsd.PSVI

Klasse **xsd.PSVI**
Die Klasse PSVI dient zur Prüfung der Schemagültigkeit von Elementen und Attributen. *Zur Beschreibung*.

Die PSVI-Klasse stellt die folgenden **Konstanten** zur Verfügung. Siehe auch xsd.ElementPSVI und xsd.AttributePSVI.

PSVI.NOTKNOWN
PSVI.VALID
PSVI.INVALID
PSVI.NONE
PSVI.FULL
PSVI.PARTIAL
PSVI.SIMPLE
PSVI.COMPLEX

Die PSVI-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

PSVI.ID_IDREF_table
Siehe xsd.ID_IDREF_table.

Die PSVI-Klasse stellt die folgenden **Instanzmethoden** zur Verfügung:

PSVI.element()
Liefert ein Element für die Prüfung der Schemagültigkeit. Siehe auch xsd.ElementPSVI.

PSVI.attribute()
Liefert ein Attribut für die Prüfung der Schemagültigkeit. Siehe auch xsd.AttributePSVI.

5.7.35 xsd.Particle

Klasse **xsd.Particle**
Eine `Particle`-Klasse enthält die Komponenten. Diese Komponenten sind entweder direkt oder indirekt enthalten. Direkt enthalten ist die Komponente, die der Wert ihres `term` Attributs ist. Indirekt enthalten sind die Partikel, Gruppen, Wildcards und Elementdeklarationen, die im Wert ihrer `term` Eigenschaft enthalten sind. *Zur Beschreibung*.

Die `Particle`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

Particle.min_occurs
Ein `xs:nonNegativeInteger`-Wert. Obligatorisch.

Particle.max_occurs
Entweder eine positive Ganzzahl oder `unbounded`. Obligatorisch.

Particle.term
Eine Term-Komponente. Obligatorisch.

Particle.annotations
Eine Sequenz von `Annotation`-Komponenten.

© 2014 Altova GmbH

5.7.36 xsd.QName

Klasse **xsd.QName**
Die Klasse QName repräsentiert einen qualifizierten XML-Namen. *Zur Beschreibung*

Die QName-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

QName.namespace_name
Der Name des Namespace-Teils des qualifizierten Namens.

QName.local_part
Der lokale Teil des qualifizierten Namens.

5.7.37 xsd.Schema

Klasse **xsd.Schema**
Die Klasse `schema` enthält eine Sammlung von Schemakomponenten, z.B. Typdefinitionen und Elementdeklarationen, die einen gemeinsamen Ziel-Namespace haben. *Zur Beschreibung*.

Die `Schema`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

`Schema.type_definitions`
Eine Gruppe von Typdefinitionen. Dabei kann es sich um eine `SimpleTypeDefinition` oder eine `ComplexTypeDefinition` handeln.

`Schema.attribute_declarations`
Eine Gruppe von `AttributeDeclaration`-Komponenten.

`Schema.element_declarations`
Eine Gruppe von `ElementDeclaration`-Komponenten.

`Schema.attribute_group_definitions`
Eine Gruppe von `AttributeGroupDefinition`-Komponenten.

`Schema.model_group_definitions`
Eine Gruppe von `ModelGroupDefinition`-Komponenten

`Schema.notation_declarations`
Eine Gruppe von `NotationDeclaration`-Komponenten.

`Schema.identity_constraint_definitions`
Eine Gruppe von `IdentityConstraintDefinition`-Komponenten.

Die `Schema`-Klasse stellt die folgenden **Instanzmethoden** zur Verfügung:

`Schema.resolve_type_definition()`
Stellt Typdefinitionen bereit.

`Schema.resolve_attribute_declaration()`
Stellt Attributdefinitionen bereit.

`Schema.resolve_element_declaration()`
Stellt Elementdeklarationen bereit.

`Schema.resolve_attribute_group_definition()`
Stellt Attributgruppendefinitionen bereit.

`Schema.resolve_model_group_definition()`
Stellt Elementgruppendefinitionen bereit.

`Schema.resolve_notation_declaration()`
Stellt Notationsdeklarationen bereit.

 © 2014 Altova GmbH

`Schema.resolve_identity_constraint_definition()`
Stellt Identity Constraint-Definitionen bereit.

5.7.38 xsd.Scope

Klasse **xsd.Scope**
Ein `Scope` Eigenschaftseintrag. Obligatorisch. *Zur Beschreibung*

Die `Scope`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

Scope.GLOBAL
Scope.LOCAL

Die `Scope`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

Scope.variety
Eines von `{global, local}`. Obligatorisch.

Scope.parent
Entweder eine `ComplexTypeDefinition` oder eine `AttributeGroupDefinition`. Obligatorisch, wenn `variety local` ist, darf andernfalls nicht vorhanden sein.

 © 2014 Altova GmbH

5.7.39 xsd.Sibling

Klasse **xsd.Sibling**
Die Klasse `Sibling` repräsentiert ein Keyword Member der für das `disallowed_names` Attribut von `NamespaceConstraint` zulässigen Wertegruppe. *Zur Beschreibung*.

Die `Sibling`-Klasse stellt die folgenden **Instanzmethoden** zur Verfügung:

Sibling.__str__()

5.7.40 xsd.SimpleTypeDefinition

Klasse **xsd.SimpleTypeDefinition**
Die Klasse `SimpleTypeDefinition` repräsentiert durch ihr `name`- und `target namespace`-Attribut identifizierte simpleTypes. *Zur Beschreibung*.

Die `SimpleTypeDefinition`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

SimpleTypeDefinition.ATOMIC
SimpleTypeDefinition.LIST
SimpleTypeDefinition.UNION

Die `SimpleTypeDefinition`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

SimpleTypeDefinition.annotations
Eine Sequenz von `Annotation`-Komponenten.

SimpleTypeDefinition.name
Ein `xs:NCName`-Wert. Optional.

SimpleTypeDefinition.target_namespace
Ein `xs:anyURI`-Wert. Optional.

SimpleTypeDefinition.context
Obligatorisch, wenn das Instanzattribut `name` (*siehe oben*) fehlt. Darf andernfalls nicht vorhanden sein. Entweder eine `AttributeDeclaration`, `ElementDeclaration`, `ComplexTypeDefinition` oder eine `SimpleTypeDefinition`.

SimpleTypeDefinition.base_type_definition
Eine Typdefinitionskomponente. Obligatorisch.

SimpleTypeDefinition.facets
Eine Menge einschränkender Facet-Komponenten.

SimpleTypeDefinition.final
Eine Untermenge von `{extension, restriction, list, union}`.

SimpleTypeDefinition.variety
Eines von `{atomic, list, union}`. Für alle simpleType-Definitionen mit Ausnahme von `xs:anySimpleType`, in der es fehlt, obligatorisch.

SimpleTypeDefinition.primitive_type_definition
Eine simpleType-Definition-Komponente. Mit einer einzigen Ausnahme obligatorisch, wenn `variety` atomar ist, darf andernfalls nicht vorhanden sein. Die Ausnahme bildet `xs:anyAtomicType`, dessen `primitive_type_definition` nicht vorhanden ist. Falls eine vorhanden ist, muss es eine primitive Definition sein.

SimpleTypeDefinition.item_type_definition

 © 2014 Altova GmbH

Eine simpleType-Definition-Komponente. Obligatorisch, wenn `variety list` ist, darf andernfalls nicht vorhanden sein. Der Wert dieser Eigenschaft muss eine primitive oder gewöhnliche simpleType Definition mit `variety=atomic` sein oder eine gewöhnliche simpleType-Definition mit `variety=union`, deren Basismitglieder atomar sind; der Wert darf nicht selbst ein Listentyp (`variety=list`) sein oder Basismitglieder vom Listentyp enthalten.

`SimpleTypeDefinition.member_type_definitions`

Eine Sequenz von primitiven oder gewöhnlichen `SimpleTypeDefinition`-Komponenten. Muss vorhanden sein (darf aber leer sein), wenn `variety=union`, darf andernfalls nicht vorhanden sein. Die Sequenz kann jede primitive oder gewöhnliche Typdefinition enthalten, darf aber keine speziellen Typdefinitionen enthalten.

5.7.41 xsd.TypeAlternative

Klasse **xsd.TypeAlternative**

Die Klasse `TypeAlternative` wird von einer `ElementDeclaration` zur Definition einer Bedingung (`test`) verwendet, unter der ein bestimmter Typ (`type_definition`) als geltende Typdefinition für Informationseinheiten für Elemente, für die diese `ElementDeclaration` gilt, verwendet wird. Jede `ElementDeclaration` kann in ihrer Typtabelle (`TypeTable`) mehrere Typalternativen haben. *Zur Beschreibung*

Die `TypeAlternative`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

TypeAlternative.annotations
Eine Sequenz von `Annotation`-Komponenten.

TypeAlternative.test
Ein `XPathExpression` Eigenschaftseintrag, mit dem eine Bedingung für die Auswahl der geltenden Typdeklaration einer Elementdeklaration definiert wird. Optional.

TypeAlternative.type_definition
Eine Typdefinitionskomponente (`xsd.ComplexTypeDefnition` oder `xsd.SimpleTypeDefinition`). Obligatorisch.

 © 2014 Altova GmbH

5.7.42 xsd.TypeTable

Klasse **xsd.TypeTable**
Die Typdefinition, anhand welcher eine Informationseinheit für Elemente validiert wird (die geltende Typdefinition), kann eine andere sein, als die deklarierte Typdefinition. Aufgrund der `TypeTable`-Eigenschaft einer `ElementDeclaration`, die die Zuweisung von Typen auf Basis von Bedingungen regelt, und aufgrund des `xsi:type`-Attributs einer Informationseinheit für Elemente können die geltende Typdefinition und die deklarierte Typdefinition unterschiedlich sein. *Zur Beschreibung*.

Die `TypeTable`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

TypeTable.alternatives
Eine Sequenz von `TypeAlternative`-Komponenten.

TypeTable.default_type_definition
Eine `TypeAlternative`-Komponente. Obligatorisch.

5.7.43 xsd.Unbounded

Klasse **xsd.Unbounded**
Die Klasse `Unbounded` ist ein String-Wert. Sie repräsentiert den oberen Wert der Eigenschaft `maxOccurs`. *Zur Beschreibung.*

Die `Unbounded`-Klasse stellt die folgenden **Instanzmethoden** zur Verfügung:

Unbounded.__str__()

© 2014 Altova GmbH

5.7.44 xsd.ValueConstraint

Klasse **xsd.ValueConstraint**
Die Klasse `ValueConstraint` repräsentiert eine Eigenschaft der Klasse `AttributeUse`. *Zur Beschreibung*.

Die `ValueConstraint`-Klasse stellt die folgenden **Konstanten** zur Verfügung:

ValueConstraint.DEFAULT
ValueConstraint.FIXED

Die `ValueConstraint`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

ValueConstraint.variety
Eines von `{default, fixed}`. Obligatorisch.

ValueConstraint.value
Ein tatsächlicher Wert. Obligatorisch.

ValueConstraint.lexical_form
Ein Zeichenstring. Obligatorisch.

5.7.45 xsd.XPathExpression

Klasse **xsd.XPathExpression**
Um eine Assertion zu überprüfen, wird eine Instanz des XPath 2.0-Datenmodells erzeugt, in der die überprüfte Informationseinheit für Elemente der Root Node (der oberste Node) ist und den Elementen und Attributen Typen und Werte gemäß den Regeln des XPath 2.0-Datenmodells zugewiesen werden. Bei Auswertung anhand dieser Datenmodelltestinstanz ergibt die Auswertung von `test` entweder `True` oder `False`. *Zur Beschreibung*.

Die `XPathExpression`-Klasse stellt die folgenden (schreibgeschützten) **Instanzattribute** zur Verfügung:

XPathExpression.namespace_bindings
Eine Gruppe von `NamespaceBinding`-Eigenschaftseinträgen für den XPath-Ausdruck.

XPathExpression.default_namespace
Ein `xs:anyURI`-Wert. Optional.

XPathExpression.base_URI
Ein `xs:anyURI`-Wert. Optional. Die Basis-URI für relative Pfade im XPath-Ausdruck.

XPathExpression.expression
Ein XPath 2.0-Ausdruck. Obligatorisch.

 © 2014 Altova GmbH

5.7.46 Spezielle vordefinierte Datentypobjekte

Es stehen die folgenden speziellen vordefinierten Datentypobjekte zur Verfügung. Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation in den Abschnitten Special Built-in Datatypes und Primitive Datatypes.

Klasse **xsd.anyAtomicType**
Eine `anyAtomicType`-Klasse repräsentiert eine Einschränkung von `anySimpleType` und ist der Basistyp der primitiven Typen.

Klasse **xsd.anySimpleType**
Eine `anySimpleType`-Klasse repräsentiert eine Einschränkung von `anyType` und ist der Basistyp von `anyAtomicType`.

Klasse **xsd.anyURI**
Eine `anyURI` -Klasse repräsentiert eine Internationalized Resource Identifier (IRI)-Referenz. Ihr Wert kann absolut oder relativ sein. Sie hat ein einziges schreibgeschütztes Instanzattribut: **anyURI.value**.

5.7.47 String-Datentypobjekte

Es stehen die folgenden String-Datentypobjekte zur Verfügung. Jeder davon ist mit seinen schreibgeschützten Instanzattributen aufgelistet.

Klasse	Instanzattribute (schreibgeschützt)
`xsd.language`	`language.value`
`xsd.normalizedString`	`normalizedString.value`
`xsd.string`	`string.value`
`xsd.token`	`token.value`

Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation in den Abschnitten Special Built-in Datatypes und Primitive Datatypes.

© 2014 Altova GmbH

5.7.48 Boolesches Datentypobjekt

Klasse **xsd.boolean**
Ein `boolean` Objekt repräsentiert ein XBRL-Instanzdokument. Es stellt das folgende schreibgeschützte Instanzattribut zur Verfügung: **boolean.value**, welches einen Booleschen Wert zurückgibt. Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation im Abschnitt Primitive Datatypes.

5.7.49 Objekte vom Datentyp Number

Es stehen die folgenden number-Datentypobjekte zur Verfügung. Jedes davon hat ein einziges schreibgeschütztes Instanzattribut: **`value`**, wobei die lexikalische Darstellung jedes Instanzattributs sich je nach Objekt unterscheidet.

Klasse	Instanzattribut (schreibgeschützt)
`xsd.byte`	`byte.value`
`xsd.decimal`	`decimal.value`
`xsd.double`	`double.value`
`xsd.float`	`float.value`
`xsd.int`	`int.value`
`xsd.integer`	`integer.value`
`xsd.long`	`long.value`
`xsd.negativeInteger`	`negativeInteger.value`
`xsd.nonNegativeInteger`	`nonNegativeInteger.value`
`xsd.nonPositiveInteger`	`nonPositiveInteger.value`
`xsd.positiveInteger`	`positiveInteger.value`
`xsd.short`	`short.value`
`xsd.unsignedByte`	`unsignedByte.value`
`xsd.unsignedInt`	`unsignedInt.value`
`xsd.unsignedLong`	`unsignedLong.value`
`xsd.unsignedShort`	`unsignedShort.value`

Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation in den Abschnitten Primitive Datatypes und Other Built-in Datatypes.

 © 2014 Altova GmbH

5.7.50 Objekte vom Datentyp Duration

Es stehen die folgenden Duration-Datentypobjekte zur Verfügung. Jedes davon ist mit seinen schreibgeschützten Instanzattributen aufgelistet.

Klasse	Instanzattribute (schreibgeschützt)
`xsd.dayTimeDuration`	`dayTimeDuration.months` `dayTimeDuration.seconds`
`xsd.duration`	`duration.months` `duration.seconds`
`xsd.yearMonthDuration`	`yearMonthDuration.months` `yearMonthDuration.seconds`

Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation in den Abschnitten Primitive Datatypes und Other Built-in Datatypes.

5.7.51 Objekte vom Datentyp Date und Time

Es stehen die folgenden Duration-Datentypobjekte zur Verfügung. Jedes davon ist mit seinen schreibgeschützten Instanzattributen aufgelistet. Wenn ein `value` Attribut vorhanden ist, wird es aus Fragmenten zusammengesetzt, die in Form anderer Attribute des Objekts verfügbar sind. Z.B.: `time.value` besteht aus den Fragmenten `time.hour`, `time.minute`, `time.second` und `time.timezoneOffset`.

Klasse	Instanzattribute (schreibgeschützt)
`xsd.date`	`dayTimeDuration.months` `dayTimeDuration.seconds`
`xsd.dateTime`	`duration.months` `duration.seconds`

`xsd.dateTimeStamp`	`dateTimeStamp.value`
	`dateTimeStamp.year`
	`dateTimeStamp.month`
	`dateTimeStamp.day`
	`dateTimeStamp.hour`
	`dateTimeStamp.minute`
	`dateTimeStamp.second`
	`dateTimeStamp.timezoneOffset`

`xsd.gDay`	`gDay.day` `gDay.timezoneOffset`
`xsd.gMonth`	`gMonth.month` `gMonth.timezoneOffset`

`xsd.gMonthDay`	`gMonthDay.month`
	`gMonthDay.days`
	`gMonthDay.timezoneOffset`

`xsd.gYear`	`gYear.year` `gYear.timezoneOffset`

`xsd.gYearMonth`	`gYearMonth.year`
	`gYearMonth.month`
	`gYearMonth.timezoneOffset`

`xsd.time`	`time.value`
	`time.hour`
	`time.minute`
	`time.second`
	`time.timezoneOffset`

Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation in den Abschnitten Primitive Datatypes und Other Built-in Datatypes.

© 2014 Altova GmbH

5.7.52 Objekte vom Datentyp Binary

Es stehen die folgenden Binary-Datentypobjekte zur Verfügung. Jedes davon ist mit seinen schreibgeschützten Instanzattributen aufgelistet.

Klasse	Instanzattribute (schreibgeschützt)
`xsd.base64Binary`	`base64Binary.value`
`xsd.hexBinary`	`hexBinary.value`

Eine ausführliche Beschreibung des Datentyps finden Sie in der XML-Schema-Spezifikation im Abschnitt Primitive Datatypes.

5.7.53 Facet-Objekte

Durch Restriktion abgeleitete Datentypen können auch durch die Spezifikation zugelassene einschränkende Facets haben. Es stehen die folgenden Facet-Objekte zur Verfügung. In der Tabelle sind Facet-Objekte aufgelistet, die nur schreibgeschützte Instanzattribute haben. Die Objekte `xsd.explicitTimezoneFacet` und `xsd.whiteSpaceFacet` haben zusätzlich zu ihren schreibgeschützten Instanzattributen Constraints und sind unterhalb der Tabelle aufgelistet.

Eine ausführliche Beschreibung eines Facet finden Sie in der XML-Schema-Spezifikation im Abschnitt Constraining Facets. (Durch Klick auf einen Facet-Objekt-Link gelangen Sie direkt zu seiner Beschreibung.)

Klasse	Instanzattribute (schreibgeschützt)
xsd.assertionsFacet	assertionsFacet.annotations assertionsFacet.value
xsd.enumerationFacet	enumerationFacet.annotations enumerationFacet.value
xsd.fractionDigitsFacet	fractionDigitsFacet.annotations fractionDigitsFacet.value
xsd.lengthFacet	lengthFacet.annotations lengthFacet.value lengthFacet.fixed
xsd.maxExclusiveFacet	maxExclusiveFacet.annotations maxExclusiveFacet.value maxExclusiveFacet.fixed
xsd.maxInclusiveFacet	maxInclusiveFacet.annotations maxInclusiveFacet.value maxInclusiveFacet.fixed
xsd.maxLengthFacet	maxLengthFacet.annotations maxLengthFacet.value maxLengthFacet.fixed
xsd.minExclusiveFacet	minExclusiveFacet.annotations minExclusiveFacet.value minExclusiveFacet.fixed
xsd.minInclusiveFacet	minInclusiveFacet.annotations minInclusiveFacet.value minInclusiveFacet.fixed
xsd.minLengthFacet	minLengthFacet.annotations minLengthFacet.value minLengthFacet.fixed
xsd.pattern	patternFacet.annotations patternFacet.value
xsd.totalDigitsFacet	totalDigitsFacet.annotations totalDigitsFacet.value totalDigitsFacet.fixed

xsd.explicitTimezoneFacet

- *Konstanten:*

 © 2014 Altova GmbH

```
explicitTimezoneFacet.REQUIRED
explicitTimezoneFacet.PROHIBITED
explicitTimezoneFacet.OPTIONAL
```

- *Schreibgeschützte Instanzattribute*:

```
whiteSpaceFacet.annotations
whiteSpaceFacet.value
whiteSpaceFacet.fixed
```

xsd.whiteSpaceFacet

- *Konstanten*:

```
whiteSpaceFacet.PRESERVE
whiteSpaceFacet.REPLACE
whiteSpaceFacet.COLLAPSE
```

- *Schreibgeschützte Instanzattribute*:

```
whiteSpaceFacet.annotations
whiteSpaceFacet.value
whiteSpaceFacet.fixed
```

Kapitel 6

Java-Schnittstelle

6 Java-Schnittstelle

Die RaptorXML API kann von Java-Code aus aufgerufen werden. Dazu müssen sich die unten aufgelisteten Bibliotheken im Classpath befinden. Diese Bibliotheken sind im Installationsordner im `bin`-Ordner installiert.

- `RaptorXMLServer.jar`: Die Bibliothek, die über HTTP-Requests mit dem RaptorXML Server kommuniziert.
- `RaptorXMLServer_JavaDoc.zip`: Eine Javadoc-Datei, die die Hilfedokumentation zur Java API enthält.

Note: Um die Java API verwenden zu können, muss sich die Jar-Datei im Java-Classpath befinden. Sie können die Jar-Datei in jeden beliebigen Ordner kopieren, falls dies für Ihre Projektkonfiguration besser geeignet ist als das Referenzieren der Datei vom ursprünglich installierten Pfad aus.

Überblick über die Schnittstelle
Die Java-API ist im `com.altova.raptorxml`-Paket verpackt. Die RaptorXML-Klasse stellt als Eintrittspunkt eine Methode namens `getFactory()` zur Verfügung, die `RaptorXMLFactory`-Objekte bereitstellt. Dadurch kann mit dem Aufruf: `RaptorXML.getFactory()` eine `RaptorXMLFactory`-Instanz erstellt werden.

Die `RaptorXMLFactory`-Schnittstelle enthält Methoden zum Aufrufen von Prozessorobjekten für die Validierung und weitere Verarbeitung (wie z.B. die XSLT-Transformation).

Anmerkung: Die `getFactory`-Methode gibt je nachdem, welche RaptorXML-Edition installiert ist, das entsprechende Fabrikobjekt zurück.

Die öffentliche Schnittstelle von `RaptorXMLFactory` wird durch das folgende Codefragment beschrieben:

```
public interface RaptorXMLFactory
{
   public XMLValidator getXMLValidator();
   public XQuery getXQuery();
   public XSLT getXSLT();
   public void setServerName(String name) throws RaptorXMLException;
   public void setServerFile(String file) throws RaptorXMLException;
   public void setServerPort(int port) throws RaptorXMLException;
   public void setGlobalCatalog(String catalog);
   public void setUserCatalog(String catalog);
   public void setGlobalResourcesFile(String file);
   public void setGlobalResourceConfig(String config);
   public void setErrorFormat(ENUMErrorFormat format);
```

 © 2014 Altova GmbH

```
    public void setErrorLimit(int limit);
    public void setReportOptionalWarnings(boolean report);
}
```

Nähere Informationen dazu finden Sie unter der Beschreibung zu RaptorXMLFactory und den dazugehörigen Java-Schnittstellen. Siehe auch Java-Beispielprojekt.

6.1 Java-Beispielprojekt

Im nachstehenden Java-Codefragment wird gezeigt, wie grundlegende Funktionalitäten aufgerufen werden können. Der Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- Navigation zum Ordner "examples" und Erstellen einer RaptorXML COM-Objektinstanz
- Validieren einer XML-Datei
- Durchführen einer XSLT-Transformation und Rückgabe des Ergebnisses als String
- Verarbeiten eines XQuery-Dokuments, Rückgabe des Ergebnisses als String
- Ausführen des Projekts

Diese Grundfunktionalität ist in den Dateien im RaptorXML Server-Applikationsordner im Ordner `examples/API` enthalten.

```
public class RunRaptorXML
{
    // Locate samples installed with the product
    // (will be two levels higher from examples/API/Java)
    // REMARK: You might need to modify this path
       static final String strExamplesFolder = System.getProperty("user.dir") +
"/../../" ;

       static com.altova.raptorxml.RaptorXMLFactory rxml;
```

```
       static void ValidateXML() throws com.altova.raptorxml.RaptorXMLException
       {
           com.altova.raptorxml.XMLValidator xmlValidator =
rxml.getXMLValidator();
           System.out.println("RaptorXML Java - XML validation");
           xmlValidator.setInputXMLFromText( "<!DOCTYPE root [ <!ELEMENT root
(#PCDATA)> ]> <root>simple input document</root>" );
              if( xmlValidator.isWellFormed() )
                  System.out.println( "The input string is well-formed" );
              else
                  System.out.println( "Input string is not well-formed: " +
xmlValidator.getLastErrorMessage() );

              if( xmlValidator.isValid() )
                  System.out.println( "The input string is valid" );
              else
                  System.out.println( "Input string is not valid: " +
xmlValidator.getLastErrorMessage() );
       }
```

```
       static void RunXSLT() throws com.altova.raptorxml.RaptorXMLException
       {
```

```
            System.out.println("RaptorXML Java - XSL Transformation");
            com.altova.raptorxml.XSLT xsltEngine = rxml.getXSLT();
            xsltEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
            xsltEngine.setXSLFileName( strExamplesFolder + "transform.xsl" );
            String result = xsltEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Transformation failed: " +
xsltEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }
```

```
        static void RunXQuery() throws com.altova.raptorxml.RaptorXMLException
        {
            System.out.println("RaptorXML Java - XQuery execution");
            com.altova.raptorxml.XQuery xqEngine = rxml.getXQuery();
            xqEngine.setInputXMLFileName( strExamplesFolder + "simple.xml" );
            xqEngine.setXQueryFileName( strExamplesFolder + "CopyInput.xq" );
            System result = xqEngine.executeAndGetResultAsString();
            if( result == null )
                System.out.println( "Execution failed: " +
xqEngine.getLastErrorMessage() );
            else
                System.out.println( "Result is " + result );
        }
```

```
        public static void main(String[] args)
        {
            try
            {
                rxml = com.altova.raptorxml.RaptorXML.getFactory();
                rxml.setErrorLimit( 3 );

                ValidateXML();
                RunXSLT();
                RunXQuery();
            }

            catch( com.altova.raptorxml.RaptorXMLException e )
            {
                e.printStackTrace();
            }

        }

}
```

6.2 RaptorXML-Schnittstellen für Java

Im Folgenden finden Sie eine Zusammenfassung der Java-Schnittstellen der RaptorXML-API. Eine ausführliche Beschreibung dazu finden Sie im jeweiligen Abschnitt.

- `RaptorXMLFactory`
  Erstellt über einen nativen Aufruf eine neue RaptorXML COM-Objektinstanz und ermöglicht den Zugriff auf RaptorXML-Prozessoren.
- `XMLValidator`
  Schnittstelle für den XML-Validierungsprozessor.
- `XSLT`
  Schnittstelle für die XSLT-Prozessoren.
- `XQuery`
  Schnittstelle für die XQuery-Prozessoren.
- `RaptorXMLException`
  Schnittstelle für die `RaptorXMLException`-Methode.

© 2014 Altova GmbH

6.2.1 RaptorXMLFactory

```
public interface RaptorXMLFactory
```

Beschreibung
Mit Hilfe von `RaptorXMLFactory()` wird eine neue RaptorXML COM-Objektinstanz erstellt, über die Sie Zugriff auf die RaptorXML-Prozessoren erhalten. Die Beziehung zwischen `RaptorXMLFactory` und dem RaptorXML COM-Objekt ist eine 1:1-Beziehung, d.h. nachfolgende Aufrufe der `get<ENGINENAME>()` Funktion geben Schnittstellen für dieselbe Prozessorinstanz zurück.

Zuerst sind die Methoden der `RaptorXMLFactory`-Schnittstelle beschrieben, gefolgt von ihren Enumerationen.

Methoden
Die Methoden der Klasse sind unterhalb in alphabetischer Reihenfolge beschrieben. In der Tabelle sind sie aus Gründen der Übersichtlichkeit nach Gruppen geordnet.

Prozessoren	Fehler, Warnungen
getXMLValidator	setErrorFormat
getXQuery	setErrorLimit
getXSLT	setReportOptionalWarnings

Kataloge	Globale Ressourcen	HTTP-Server
setGlobalCatalog	setGlobalResourceConfig	setServerFile
setUserCatalog	setGlobalResourcesFile	setServerName
		setServerPort

Produktinformationen	
getProductName	Is64Bit
getProductNameAndVersion	getAPIMajorVersion
getMajorVersion	getAPIMinorVersion
getMinorVersion	getAPIServicePackVersion
getServicePackVersion	

Nach oben | Methoden | Enumerationen

getAPIMajorVersion
`public int` **`getAPIMajorVersion()`**
Gibt die Hauptversion der API als Ganzzahl zurück. Die Hauptversion der API muss nicht

unbedingt mit der Hauptversion des Produkts übereinstimmen, wenn die API mit einem anderen Server verbunden ist.
Rückgabe:
eine Ganzzahl, die die Hauptversion der API ist.

Nach oben | Methoden | Enumerationen

getAPIMinorVersion
`public int getAPIMinorVersion()`
Gibt die Nebenversion der API als Ganzzahl zurück. Die Nebenversion der API muss nicht unbedingt mit der Nebenversion des Produkts übereinstimmen, wenn die API mit einem anderen Server verbunden ist.
Rückgabe:
eine Ganzzahl, die die Nebenversion der API ist.

Nach oben | Methoden | Enumerationen

getAPIServicePackVersion
`public int getAPIServicePackVersion()`
Gibt die Service Pack-Version der API als Ganzzahl zurück. Die Service Pack-Version der API muss nicht unbedingt mit der Service Pack-Version des Produkts übereinstimmen, wenn die API mit einem anderen Server verbunden ist.
Rückgabe:
eine Ganzzahl, die die Service Pack-Version der API ist.

Nach oben | Methoden | Enumerationen

getMajorVersion
`public int getMajorVersion()`
Gibt die Hauptversion des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2014r2sp1(x64)` den Wert `16` zurück (den Unterschied zwischen der Hauptversion (`2014`) und dem Anfangsjahr `1998`). Löst bei Fehler eine `RaptorXMLException` aus.
Rückgabe:
eine Ganzzahl, die die Hauptversion des Produkts ist.

Nach oben | Methoden | Enumerationen

getMinorVersion
`public int getMinorVersion()`
Gibt die Nebenversion des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `2` zurück (den Unterschied zur Nebenversionsnummer `r2`). Löst bei Fehler eine `RaptorXMLException` aus.
Rückgabe:
eine Ganzzahl, die die Nebenversion des Produkts ist.

 © 2014 Altova GmbH

Nach oben | Methoden | Enumerationen

getProductName
`public string` **getProductName()**
Gibt den Namen des Produkts als String zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `Altova RaptorXML Server` zurück. Löst bei Fehler eine `RaptorXMLException` aus.
Rückgabe:
ein String, der der Name des Produkts ist.

Nach oben | Methoden | Enumerationen

getProductNameAndVersion
`public string` **getProductNameAndVersion()**
Gibt den Namen des Produkts und seine Version als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `Altova RaptorXML Server 2015r2sp1(x64)` zurück. Löst bei Fehler eine `RaptorXMLException` aus.

Rückgabe:
ein String, der der Name und die Version des Produkts ist.

Nach oben | Methoden | Enumerationen

getServicePackVersion
`public int` **getServicePackVersion()**
Gibt die Service Pack-Version des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `RaptorXML Server 2015r2sp1(x64)` den Wert `1` (aus der Service Pack Versionsnummer `sp1`) zurück. Löst bei Fehler eine `RaptorXMLException` aus.

Rückgabe:
eine Ganzzahl, die die Service Pack-Version des Produkts ist.

Nach oben | Methoden | Enumerationen

getXMLValidator
`public XMLValidator` **getXMLValidator()**
Ruft den XMLValidator auf.
Rückgabe:
eine neue `XMLValidator`-Instanz dieser `RaptorXMLFactory`.

Nach oben | Methoden | Enumerationen

getXQuery
`public XQuery getXQuery()`
Ruft den XQuery-Prozessor auf.
Rückgabe:
eine neue `XQuery`-Instanz dieser `RaptorXMLFactory`.

Nach oben | Methoden | Enumerationen

getXSLT
`public XSLT getXSLT()`
Ruft den XSLT-Prozessor auf.
Rückgabe:
eine neue `XSLT`-Instanz dieser `RaptorXMLFactory`.

Nach oben | Methoden | Enumerationen

is64Bit
`public boolean is64Bit()`
Überprüft, ob die Applikation eine ausführbare 64-Bit-Datei ist. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `True` zurück. Löst bei Fehler eine `RaptorXMLException` aus.
Rückgabe:
Der Boolesche Wert `true`, wenn die Applikation eine 64-Bit-Applikation ist, andernfalls `false`.

Nach oben | Methoden | Enumerationen

setErrorFormat
`public void setErrorFormat(ENUMErrorFormat format)`
Definiert als das RaptorXML-Fehlerformat eines der `ENUMErrorFormat`-Literale (Text, ShortXML, LongXML).
Parameter:
`format`: Enthält den Wert des ausgewählten `ENUMErrorFormat`-Literals.

Nach oben | Methoden | Enumerationen

setErrorLimit
`public void setErrorLimit(int limit)`
Definiert das Limit für RaptorXML-Validierungsfehler.
Parameter:
`limit`: Ist vom Typ `int` und definiert die Anzahl der Fehler, die ausgegeben werden, bevor die Ausführung gestoppt wird. Mit Hilfe von `-1` können Sie `limit` auf unbegrenzt setzen (d.h. alle

Fehler werden ausgegeben). Der Standardwert ist `100`.

Nach oben | Methoden | Enumerationen

setGlobalCatalog

```
public void setGlobalCatalog(String catalog)
```

Definiert den Pfad der Hauptkatalogdatei (Eintrittspunkt) in Form einer URL.

Parameter:

`catalog`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur Hauptkatalogdatei, die verwendet werden soll, angibt.

Nach oben | Methoden | Enumerationen

setGlobalResourceConfig

```
public void setGlobalResourceConfig(String config)
```

Definiert die aktive Konfiguration der globalen Ressource.

Parameter:

`config`: Ist vom Typ `String` und gibt den Namen der von der aktiven globalen Ressource zu verwendenden Konfiguration an.

Nach oben | Methoden | Enumerationen

setGlobalResourcesFile

```
public void setGlobalResourcesFile(String file)
```

Definiert den Pfad der XML-Datei für globale Ressourcen in Form einer URL.

Parameter:

`file`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur XML-Datei für globale Ressourcen angibt.

Nach oben | Methoden | Enumerationen

setReportOptionalWarnings

```
public void setReportOptionalWarnings(boolean report)
```

Aktiviert/Deaktiviert die Ausgabe von Warnungen. Mit dem Wert `true` werden Warnungen aktiviert; mit `false` werden sie deaktiviert.

Parameter:

`report`: Akzeptiert die Booleschen Werte `true` oder `false`.

Nach oben | Methoden | Enumerationen

setServerFile

`public void setServerFile(String file)`

Definiert den Pfad zur Konfigurationsdatei des HTTP-Servers relativ zur HTTP-Server-Adresse. Verursacht eine `RaptorXMLException`, wenn ein Fehler auftritt.

Parameter:

`file`: Ein String, der die Adresse der HTTP-Server-Konfigurationsdatei relativ zur Server-Adresse angibt.

Nach oben | Methoden | Enumerationen

setServerName

`public void setServerName(String name)`

Definiert den Namen des HTTP-Servers. Verursacht eine `RaptorXMLException`, wenn ein Fehler auftritt.

Parameter:

`name`: Ein String, der den Namen des HTTP-Servers angibt.

setServerPort

`public void setServerPort(int port)`

Definiert den Port des HTTP-Servers, über den der Dienst aufgerufen wird. Es muss sich um einen festen Port handeln, damit HTTP-Requests korrekt an den Dienst adressiert werden können. Verursacht eine `RaptorXMLException`, wenn ein Fehler auftritt.

Parameter:

`port`: Eine Ganzzahl, die den Zugriffsport am HTTP-Server angibt.

Nach oben | Methoden | Enumerationen

setUserCatalog

`public void setUserCatalog(String catalog)`

Definiert den Pfad der benutzerdefinierten Katalogdatei in Form einer URL.

Parameter:

`catalog`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden benutzerdefinierten Katalogdatei angibt.

Nach oben | Methoden | Enumerationen

Enumerationen
ENUMErrorFormat

 © 2014 Altova GmbH

ENUMErrorFormat

```
public enum ENUMErrorFormat {
       eFormatText
       eFormatShortXML
       eFormatLongXML }
```

`ENUMErrorFormat` kann eines der Enumerationsliterale erhalten: `eFormatText`, `eFormatShortXML`, `eFormatLongXML`. Diese definieren das Format der Fehlermeldungen, wobei `eLongXML` die ausführlichsten Fehlermeldungen bereitstellt. Der Standardwert ist `eFormatText`.

Verwendet von (Schnittstelle::Methode):
`RaptorXMLFactory` `setErrorFormat`

Nach oben | **Methoden** | **Enumerationen**

6.2.2 XMLValidator

```
public interface XMLValidator
```

Beschreibung
Validiert das bereitgestellte XML-, Schema- oder DTD-Dokument. Die XML-Dokumentvalidierung kann anhand interner oder externer DTDs oder XML-Schemas durchgeführt werden. Überprüft auch die Wohlgeformtheit von XML-, DTD- und XML-Schema-Dokumenten. Zuerst sind die Methoden der Schnittstelle und anschließend ihre Enumerationen beschrieben.

Methoden
Die Methoden der Klasse sind nachstehend in alphabetischer Reihenfolge beschrieben. In der Tabelle sind sie aus Gründen der Übersichtlichkeit nach Gruppen geordnet.

Verarbeitung	Input-Dateien	XML-Schema
isValid(ENUM type)	setInputXMLFileName	setSchemaImports
isValid	setInputXMLFromText	setSchemalocationHints
isWellFormed(ENUM type)	setInputXMLFileCollection	setSchemaMapping
isWellFormed	setInputXMLTextCollection	setXSDVersion
getLastErrorMessage	setSchemaFileName	
setAssessmentMode	setSchemaFromText	**XML**
setPythonScriptFile	setSchemaFileCollection	setEnableNamespaces
setStreaming	setSchemaTextCollection	setXincludeSupport
	setDTDFileName	setXMLValidationMode
	setDTDFromText	

Nach oben | Methoden | Enumerationen

getLastErrorMessage
```
public String getLastErrorMessage()
```
Ruft die letzte Fehlermeldung aus dem XML-Validierungsprozessor auf.
Rückgabe:
ein String, der die letzte Fehlermeldung aus dem XML-Validierungsprozessor ist.

Nach oben | Methoden | Enumerationen

isValid

```
public boolean isValid(ENUMValidationType type)
```
Gibt das Ergebnis der Validierung des XML-Dokuments, Schema-Dokuments oder DTD-Dokuments zurück. Welcher Dokumenttyp validiert wird, wird durch den Parameter `type` definiert, der ein `ENUMValidationType` Literal als Wert erhält. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Wenn ein Fehler auftritt, wird eine `RaptorXMLException` ausgegeben. Mit Hilfe der `getLastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.
Parameter:
`type`: Ein `ENUMValidationType` Literal, welches definiert, ob ein XML-Schema, eine DTD oder ein XML-Dokument oder ein XML-Dokument anhand eines XML-Schemas oder anhand einer DTD validiert wird.
Rückgabe:
Bei Erfolg der Boolesche Wert `true`, bei Fehlschlag `false`.

Nach oben | Methoden | Enumerationen

isValid
```
public boolean isValid()
```
Gibt das Ergebnis der Validierung des angegebenen Dokuments zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
Rückgabe:
Bei Erfolg der Boolesche Wert `true`, bei Fehlschlag `false`.

Nach oben | Methoden | Enumerationen

isWellFormed
```
public boolean isWellFormed(ENUMWellformedCheckType type)
```
Gibt das Ergebnis der Wohlgeformtheitsprüfung des XML- oder DTD-Dokuments zurück. Welcher Dokumenttyp überprüft wird, wird durch den Parameter `type` angegeben. Dieser Parameter erhält ein `ENUMWellformedCheckType` Literal als Wert. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei einem Fehler wird eine `RaptorXMLException` ausgegeben. Mit Hilfe der `getLastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.
Parameter:
`type`: Ein `ENUMWellformedCheckType` Literal, welches angibt, ob ein XML-Dokument oder eine DTD auf Wohlgeformtheit geprüft wird.
Rückgabe:
Bei Erfolg der Boolesche Wert `true`, bei Fehlschlag `false`.

Nach oben | Methoden | Enumerationen

isWellFormed
```
public boolean isWellFormed()
```
Gibt das Ergebnis der Wohlgeformtheitsprüfung des XML- oder DTD-Dokuments zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
Rückgabe:
Bei Erfolg der Boolesche Wert `true`, bei Fehlschlag `false`.

Nach oben | Methoden | Enumerationen

setAssessmentMode

```
public void setAssessmentMode(ENUMAssessmentMode mode)
```

Definiert den Validierungsmodus (`Strict/Lax`), der im `mode` Parameter definiert ist, der ein `ENUMAssessmentMode` Literal erhält.

Parameter:

`mode`: Ein `ENUMAssessmentMode` Literal, das definiert, ob die Validierung streng oder lax sein oder übersprungen werden soll.

Nach oben | Methoden | Enumerationen

setDTDFileName

```
public void setDTDFileName(String filePath)
```

Definiert den Pfad des für die Validierung zu verwendenden DTD-Dokuments in Form einer URL.

Parameter:

`filePath`: Der angegebene String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden DTD definiert.

Nach oben | Methoden | Enumerationen

setDTDFromText

```
public void setDTDFromText(String dtdText)
```

Liefert den Inhalt des DTD-Dokuments als Text.

Parameter:

`dtdText`: Der bereitgestellte String ist das für die Validierung zu verwendende DTD-Dokument.

Nach oben | Methoden | Enumerationen

setEnableNamespaces

```
public void setEnableNamespaces(boolean enable)
```

Aktiviert die Namespace-fähige Verarbeitung. Dies ist nützlich, um die XML-Instanz auf Fehler infolge falscher Namespaces zu überprüfen. Der Wert `true` aktiviert die Namespace-fähige Verarbeitung; `false` deaktiviert sie. Der Standardwert ist `false`.

Parameter:

`support`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

 © 2014 Altova GmbH

setInputXMLFileCollection

```
public void setInputXMLFileCollection(Collection<?> fileCollection)
```

Definiert die Sammlung der XML-Dateien, die als Input-Daten verwendet werden. Die Dateien werden durch ihre URLs identifiziert.

Parameter:

`fileCollection`: Eine Sammlung von Strings, wobei jeder der Strings die absolute URL einer XML-Input-Datei ist.

Nach oben | Methoden | Enumerationen

setInputXMLFileName

```
public void setInputXMLFileName(String filePath)
```

Definiert den Pfad des zu validierenden XML-Dokuments in Form einer URL.

Parameter:

`filePath`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der XML-Datei angibt.

Nach oben | Methoden | Enumerationen

setInputXMLFromText

```
public void setInputXMLFromText(String inputText)
```

Liefert den Inhalt des zu validierenden XML-Dokuments

Parameter:

`inputText`: Der bereitgestellte String ist der Inhalt des zu validierenden XML-Dokuments.

Nach oben | Methoden | Enumerationen

setInputXMLTextCollection

```
public void setInputXMLTextCollection(Collection<?> stringCollection)
```

Liefert den Inhalt mehrerer XML-Dateien, die als Input-Daten verwendet werden.

Parameter:

`stringCollection`: Eine Sammlung von Strings, wobei jeder der Strings der Inhalt einer XML-Input-Datei ist.

Nach oben | Methoden | Enumerationen

setParallelAssessment

```
public void setParallelAssessment(boolean support)
```

Aktiviert oder deaktiviert die Verwendung einer parallelen Validierung. Mit `true` wird die parallele Validierung aktivert, mit `false` wird sie deaktiviert. Der Standardwert ist `false`.

Parameter:

`support`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

setPythonScriptFile

```
public void setPythonScriptFile(String file)
```

Definiert den Pfad der Python Script-Datei in Form einer URL.

Parameter:

`file`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der Python-Datei angibt.

Nach oben | Methoden | Enumerationen

setSchemaFileCollection

```
public void setSchemaFileCollection(Collection<?> fileCollection)
```

Liefert die Sammlung der XML-Dateien, die als externe XML-Schemas verwendet werden. Die Dateien werden anhand ihrer URLs identifiziert.

Parameter:

`fileCollection`: Eine Sammlung von Strings, von denen jeder die absolute URL einer XML-Schema-Datei ist.

Nach oben | Methoden | Enumerationen

setSchemaFileName

```
public void setSchemaFileName(String filePath)
```

Definiert den Pfad des zu verwendenden XML-Schema-Dokuments in Form einer URL.

Parameter:

`filePath`: Der gelieferte String muss eine absolute URL sein, die den genauen Pfad der XML-Schema-Datei angibt.

Nach oben | Methoden | Enumerationen

setSchemaFromText

```
public void setSchemaFromText(String schemaText)
```

Liefert den Inhalt des zu verwendenden XML-Schema-Dokuments.

Parameter:

`schemaText`: Der gelieferte String ist der Inhalt des zu verwendenden XML-Schema-Dokuments.

Nach oben | Methoden | Enumerationen

 © 2014 Altova GmbH

setSchemaImports

```
public void setSchemaImports(ENUMSchemaImports opt)
```

Definiert, wie Schemaimporte auf Basis der Attributwerte der `xs:import` Elemente zu behandeln sind. Die Art der Behandlung wird durch das ausgewählte `ENUMSchemaImports` Literal definiert.

Parameter:

`opt`: Enthält das `ENUMSchemaImports` Literal, das die Behandlung von Schemaimporten definiert. Nähere Informationen dazu siehe Beschreibung zu `ENUMSchemaImports`.

Nach oben | Methoden | Enumerationen

setSchemalocationHints

```
public void setSchemalocationHints(ENUMLoadSchemalocation opt)
```

Definiert, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll. Der Mechanismus wird durch das ausgewählte `ENUMLoadSchemalocation` Literal definiert.

Parameter:

`opt`: Enthält das `ENUMLoadSchemalocation` Literal, das definiert, welcher Mechanismus zum Auffinden des Schemas verwendet wird. Nähere Informationen dazu siehe Beschreibung zu `ENUMLoadSchemalocation`.

Nach oben | Methoden | Enumerationen

setSchemaMapping

```
public void setSchemaMapping(ENUMSchemaMapping opt)
```

Definiert, welches Mapping zum Auffinden des Schemas verwendet werden soll. Das Mapping wird durch das ausgewählte `ENUMSchemaMapping` Literal definiert.

Parameter:

`opt`: Enthält das `ENUMSchemaMapping` Literal. Nähere Informationen dazu siehe Beschreibung zu `ENUMSchemaMapping`.

Nach oben | Methoden | Enumerationen

setInputSchemaTextCollection

```
public void setInputSchemaTextCollection(Collection<?> stringCollection)
```

Liefert den Inhalt mehrerer XML-Schema-Dokumente.

Parameter:

`stringCollection`: Eine Sammlung von Strings, von denen jede der Inhalt eines XML-Schema-Dokuments ist.

Nach oben | Methoden | Enumerationen

setStreaming

```
public void setStreaming(boolean support)
```

Aktiviert die Streaming-Validierung. Im Streaming-Modus werden möglichst wenige Daten im

Arbeitsspeicher behalten, wodurch die Verarbeitung beschleunigt wird.
Parameter:
support: Ein Wert true aktiviert das Streaming; false deaktiviert es. Der Standardwert ist true.

Nach oben | Methoden | Enumerationen

setXincludeSupport
public void **setXIncludeSupport**(boolean support)
Aktiviert oder deaktiviert die Verwendung von XInclude Elementen. Der Wert true aktiviert die XInclude-Unterstützung; false deaktiviert sie. Der Standardwert ist false.
Parameter:
support: Erhält den Booleschen Wert true oder false.

Nach oben | Methoden | Enumerationen

setXMLValidationMode
public void **setXMLValidationMode**(ENUMXMLValidationMode mode)
Definiert den XML-Validierungsmodus, welcher ein Enumerationsliteral von ENUMXMLValidationMode ist.
Parameter:
mode: Ist ein Enumerationsliteral von ENUMXMLValidationMode, welches festlegt, ob die Gültigkeit oder Wohlgeformtheit geprüft wird.

Nach oben | Methoden | Enumerationen

setXSDVersion
public void **setXSDVersion**(ENUMXSDVersion version)
Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert wird.
Parameter:
version: Ist ein Enumerationsliteral von ENUMXSDVersion, welches die XML-Schema-Version definiert.

Nach oben | Methoden | Enumerationen

Enumerationen
ENUMAssessmentMode
ENUMLoadSchemalocation
ENUMSchemaImports
ENUMSchemaMapping

 © 2014 Altova GmbH

ENUMValidationMode
ENUMValidationType
ENUMWellformedCheckType
ENUMXSDVersion

Nach oben | Methoden | Enumerationen

ENUMAssessmentMode

```
public enum ENUMAssessmentMode {
       eAssessmentModeLax
       eAssessmentModeStrict   }
```

ENUMAssessmentMode erhält eines der Enumerationsliterale: eAssessmentModeLax, eAssessmentModeStrict. Diese definieren, ob eine strenge oder laxe Validierung erfolgen soll.

Verwendet von (Schnittstelle::Methode):

XMLValidator	setAssessmentMode

Nach oben | Methoden | Enumerationen

ENUMLoadSchemalocation

```
public enum ENUMLoadSchemalocation {
       eLoadBySchemalocation
       eLoadByNamespace
       eLoadCombiningBoth
       eLoadIgnore   }
```

ENUMLoadSchemalocation enthält das Enumerationsliteral, das den Mechanismus zum Auffinden des Schemas definiert. Die Auswahl erfolgt auf Basis des schema location-Attributs des XML-Instanzdokuments. Dieses Attribut könnte xsi:schemaLocation oder xsi:noNamespaceSchemaLocation sein.

- eLoadBySchemalocation verwendet die URL des schema location-Attributs im XML-Instanzdokument. Dieses Enumerationsliteral ist der **Standardwert**.
- eLoadByNamespace verwendet den Namespace-Teil von xsi:schemaLocation und einen leeren String im Fall von xsi:noNamespaceSchemaLocation, um das Schema über ein Katalogmapping zu finden.
- eLoadCombiningBoth: Wenn entweder die Namespace-URL oder die Schemapfad-URL ein Katalogmapping hat, so wird das Katalogmapping verwendet. Wenn beide Katalogmappings haben, hängt es vom Wert von ENUMSchemaMapping ab, welches Mapping verwendet wird. Wenn weder der Namespace noch der Schemapfad ein Katalogmapping hat, wird die Schemapfad-URL verwendet.
- eLoadCombiningBoth: Sowohl das xsi:schemaLocation Attribut als auch das xsi:noNamespaceSchemaLocation Attribut wird ignoriert.

Verwendet von (Schnittstelle::Methode):

XMLValidator	setSchemalocationHints
XSLT	setSchemalocationHints

Nach oben | Methoden | Enumerationen

ENUMSchemaImports

```
public enum ENUMSchemaImports {
        eSILoadBySchemalocation
        eSILoadPreferringSchemalocation
        eSILoadByNamespace
        eSILoadCombiningBoth
        eSILicenseNamespaceOnly    }
```

`ENUMSchemaImports` enthält das Enumerationsliteral, das das Verhalten der `xs:import` Elemente des Schemas definiert, von denen jedes ein optionales `namespace` Attribut und ein optionales `schemaLocation` Attribut hat.

- `eSILoadBySchemalocation` verwendet den Wert des `schemaLocation` Attributs zum Auffinden des Schemas, wobei Katalogmappings berücksichtigt werden. Wenn das `namespace` Attribut vorhanden ist, wird der Namespace importiert (lizenziert).
- `eSILoadPreferringSchemalocation`: Wenn das `schemaLocation` Attribut vorhanden ist, wird es verwendet, wobei Katalogmappings berücksichtigt werden. Wenn kein `schemaLocation` Attribut vorhanden ist, wird der Wert des `namespace` Attributs über ein Katalogmapping verwendet. Dieses Enumerationsliteral ist der **Standardwert.**
- `eSILoadByNamespace` verwendet den Wert des `namespace` Attributs, um das Schema über ein Katalogmapping zu finden.
- `eSILoadCombiningBoth`: Wenn entweder die `namespace` URL oder die `schemaLocation` URL ein Katalogmapping hat, wird das Katalogmapping verwendet. Wenn beide Katalogmappings haben, hängt es vom Wert von `ENUMSchemaMapping` ab, welches Mapping verwendet wird. Wenn weder die `namespace` noch die `schemaLocation` URL ein Katalogmapping hat, wird die `schemaLocation` URL verwendet.
- `eSILicenseNamespaceOnly`: Der Namespace wird importiert. Es wird kein Schema-Dokument importiert.

Verwendet von (Schnittstelle::Methode):

XMLValidator	setSchemaImports
XSLT	setSchemaImports

Nach oben | Methoden | Enumerationen

ENUMSchemaMapping

```
public enum ENUMSchemaMapping {
        eSMPreferSchemalocation
        eSMPreferNamespace    }
```

`ENUMSchemaMapping` enthält das Enumerationsliteral, das angibt, ob der Namespace oder der Schemapfad ausgewählt wird.

- `eSMPreferNamespace`: Wählt den Namespace aus.
- `eSMPreferSchemalocation`: Wählt den Schemapfad aus. Dies ist der Standardwert.

 © 2014 Altova GmbH

Verwendet von (Schnittstelle::Methode):

XMLValidator	setSchemaMapping
XSLT	setSchemaMapping

Nach oben | Methoden | Enumerationen

ENUMXMLValidationMode

```
public enum ENUMXMLValidationMode {
       eProcessingModeValid
       eProcessingModeWF  }
```

ENUMXMLValidationMode enthält das Enumerationsliteral, das angibt, welche Art von XML-Validierung durchgeführt werden soll (Validierung oder Wohlgeformtheitsprüfung).

- eProcessingModeValid: Setzt den XML-Verarbeitungsmodus auf validation.
- eProcessingModeValid: Setzt den XML-Verarbeitungsmodus auf wellformed. Dies ist der Standardwert.

Verwendet von (Schnittstelle::Methode):

XMLValidator	setXMLValidationMode
XSLT	setXMLValidationMode
XQuery	setXMLValidationMode

Nach oben | Methoden | Enumerationen

ENUMValidationType

```
public enum ENUMValidationType {
       eValidateAny
       eValidateXMLWithDTD
       eValidateXMLWithXSD
       eValidateDTD
       eValidateXSD }
```

ENUMValidationType enthält des Enumerationsliteral, das angibt, welche Validierung durchgeführt werden soll - und im Fall von XML-Dokumenten - ob das Dokument anhand einer DTD oder einer XSD-Datei validiert werden soll.

- eValidateAny: Der Dokumenttyp wird automatisch ermittelt.
- eValidateXMLWithDTD: Validiert ein XML-Dokument anhand einer DTD.
- eValidateXMLWithXSD: Validiert ein XML-Dokument anhand einer XSD-Datei (XML-Schema).
- eValidateDTD: Validiert ein DTD-Dokument.
- eValidateXSD: Validiert ein XSD-Dokument.

Verwendet von (Schnittstelle::Methode):

XMLValidator	isValid

Nach oben | Methoden | Enumerationen

ENUMWellformedCheckType

```
public enum ENUMWellformedCheckType {
        eWellformedAny
        eWellformedXML
        eWellformedDTD    }
```

ENUMWellformedCheckType enthält das Enumerationsliteral, das angibt, welche Art der Wohlgeformtheitsprüfung durchgeführt werden soll (für XML- oder DTD-Dokumente).

- eWellformedAny: Der Dokumenttyp wird automatisch ermittelt.
- eWellformedXML: Überprüft ein XML-Dokument auf Wohlgeformtheit.
- eWellformedDTD: Überprüft ein DTD-Dokument auf Wohlgeformtheit.

Verwendet von (Schnittstelle::Methode):

XMLValidator	isWellformed

Nach oben | Methoden | Enumerationen

ENUMXSDVersion

```
public enum ENUMXSDVersion {
        eXSDVersionAuto
        eXSDVersion10
        eXSDVersion11    }
```

ENUMXSDVersion enthält das Enumerationsliteral, das die XML-Schema-Version angibt.

- eXSDVersionAuto: Die XML-Schema-Version wird automatisch anhand des vc:minVersion Attributs des XSD-Dokuments ermittelt. Wenn das vc:minVersion Attribut des XSD-Dokuments den Wert 1.1 hat, wird das Dokument als XSD 1.1 erkannt. Wenn das Attribut einen anderen Wert hat oder fehlt, wird das Dokument als XSD 1.0 erkannt.
- eXSDVersion10: Setzt die XML-Schema-Version für die Validierung auf XML Schema 1.0.
- eXSDVersion11: Setzt die XML-Schema-Version für die Validierung auf XML Schema 1.1.

Verwendet von (Schnittstelle::Methode):

XMLValidator	setXSDVersion
XSLT	setXSDVersion
XQuery	setXSDVersion

Nach oben | Methoden | Enumerationen

 © 2014 Altova GmbH

6.2.3 XSLT

```
public interface XSLT
```

Beschreibung
Transformiert XML anhand des bereitgestellten XSLT 1.0-, 2.0- oder 3.0-Dokuments. XML- und XSLT-Dokumente können als Datei (über eine URL) oder als Textstring bereitgestellt werden. Die Ausgabe wird als Datei (unter dem angegebenen Pfad) oder als Textstring zurückgegeben. Es können XSLT-Parameter bereitgestellt werden und für Spezialverarbeitungen können Altova-Erweiterungsfunktionen wie z.B. für Diagramme aktiviert werden. Das XSLT-Dokument kann auch validiert werden. Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden.

Zuerst sind die Methoden der Schnittstelle und anschließend ihre Enumerationen beschrieben.

Methoden
Die Methoden der Klasse sind nachstehend in alphabetischer Reihenfolge beschrieben. In der Tabelle sind sie aus Gründen der Übersichtlichkeit nach Gruppen geordnet.

Verarbeitung	XSLT
isValid	setVersion
execute	setXSLFileName
executeAndGetResultAsString	setXSLFromText
executeAndGetResultAsStringWithBaseOutputURI	setaddExternalParameter
getLastErrorMessage	setclearExternalParametersList
setIndentCharacters	setInitialTemplateMode
setStreamingSerialization	setNamedTemplateEntryPoint

XML-Schema	XML	Erweiterungen
setSchemaImports	setInputXMLFileName	setChartExtensionsEnabled
setSchemalocationHints	setInputXMLFromText	setDotNetExtensionsEnabled
setSchemaMapping	setLoadXMLWithPSVI	setJavaExtensionsEnabled
setXSDVersion	setXincludeSupport	setJavaBarcodeExtensionLocation
	setXMLValidationMode	

Nach oben | Methoden | Enumerationen

addExternalParameter

```
public void addExternalParameter(String name, String value)
```
Fügt den Namen und Wert eines neuen externen Parameters hinzu. Jeder externe Parameter und sein Wert müssen in einem separaten Aufruf der Methode definiert werden. Die Parameter müssen im XSLT-Dokument deklariert sein. Da Parameterwerte XPath-Ausdrücke sind, müssen Parameterwerte, die Strings sind, in einfache Anführungszeichen gesetzt werden.
Parameter:
`name`: Enthält den Namen des Parameters in Form eines QName als String.
`value`: Enthält den Wert des Parameters als String.

Nach oben | Methoden | Enumerationen

clearExternalParameterList

```
public void clearExternalVariableList()
```
Löscht die mit der Methode `AddExternalParameter` erstellte Liste der externen Parameter.

Nach oben | Methoden | Enumerationen

execute

```
public boolean execute(String outputFile)
```
Führt die XSLT-Transformation anhand der in `ENUMXSLTVersion` (siehe `setVersion` Methode) genannten XSLT-Spezifikation aus und speichert das Ergebnis in der im `outputFile` Parameter genannten Ausgabedatei. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `getLastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.
Parameter:
`outputFile`: Ein String, der den Pfad (Pfad und Dateinamen) der Ausgabedatei liefert.
Rückgabe:
Bei erfolgreicher Ausführung wird der Boolesche Wert `true`, bei Fehlschlag `false` zurückgegeben.

Nach oben | Methoden | Enumerationen

executeAndGetResultAsString

```
public String executeAndGetResultAsString()
```
Führt die XSLT-Transformation anhand der in `ENUMXSLTVersion` (siehe `setVersion` Methode) genannten XSLT-Spezifikation aus und gibt das Ergebnis als String zurück. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `getLastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.
Rückgabe:
ein String, der das Ergebnis der XSLT-Transformation ist.

Nach oben | Methoden | Enumerationen

 © 2014 Altova GmbH

executeAndGetResultAsStringWithBaseOutputURI

```
public String executeAndGetResultAsStringWithBaseOutputURI(String baseURI)
```

Führt die XSLT-Transformation anhand der in ENUMXSLTVersion (siehe setVersion Methode) genannten XSLT-Spezifikation aus und gibt das Ergebnis unter dem durch die Basis-URI definierten Pfad als String zurück. Bei Auftreten eines Fehlers wird eine RaptorXMLException ausgelöst. Mit Hilfe der getLastErrorMessage Methode können Sie zusätzliche Informationen aufrufen.

Parameter:

baseURI: Ein String, der eine URI liefert.

Rückgabe:

ein String, der das Ergebnis der XSLT-Transformation ist.

Nach oben | Methoden | Enumerationen

getLastErrorMessage

```
public String getLastErrorMessage()
```

Ruft die letzte Fehlermeldung vom XSLT-Prozessor ab.

Rückgabe:

ein String, der die letzte Fehlermeldung vom XSLT-Prozessor repräsentiert.

Nach oben | Methoden | Enumerationen

isValid

```
public boolean isValid()
```

Gibt das Ergebnis der Validierung des XSLT-Dokuments, die anhand der in ENUMXSLTVersion (siehe setVersion Methode) genannten XSLT-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg true, bei Fehlschlag false. Bei Auftreten eines Fehlers wird eine RaptorXMLException ausgelöst. Mit Hilfe der getLastErrorMessage Methode können Sie zusätzliche Informationen aufrufen.

Rückgabe:

Boolescher Wert true bei Erfolg, false bei Fehlschlag.

Nach oben | Methoden | Enumerationen

setChartExtensionsEnabled

```
public void setChartExtensionsEnabled(boolean enable)
```

Aktiviert bzw. deaktiviert Altova-Diagrammerweiterungsfunktionen.

Parameter:

enable: Ein Wert true aktiviert Diagrammerweiterungen; false deaktiviert sie. Der Standardwert ist true.

Nach oben | Methoden | Enumerationen

setDotNetExtensionsEnabled

```
public void setDotNetExtensionsEnabled(boolean enable)
```

Aktiviert oder deaktiviert .NET-Erweiterungsfunktionen.

Parameter:

`enable`: Ein Wert `true` aktiviert .NET-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

Nach oben | Methoden | Enumerationen

setJavaBarcodeExtensionLocation

```
public void setJavaBarcodeExtensionLocation(String path)
```

Definiert den Pfad der Java Barcode-Erweiterungsdatei. Nähere Informationen dazu finden Sie im Abschnitt Barcode-Erweiterungsfunktionen von Altova.

Parameter:

`path`: Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden Datei angibt.

Nach oben | Methoden | Enumerationen

setJavaExtensionsEnabled

```
public void setJavaExtensionsEnabled(boolean enable)
```

Aktiviert oder deaktiviert Java-Erweiterungsfunktionen.

Parameter:

`enable`: Ein Wert `true` aktiviert Java-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

Nach oben | Methoden | Enumerationen

setIndentCharacters

```
public void setIndentCharacters(String chars)
```

Definiert den in der Ausgabe als Einrückung zu verwendenden Zeichenstring.

Parameter:

`chars`: Enthält den Einrückungszeichenstring.

Nach oben | Methoden | Enumerationen

setInitialTemplateMode

```
public void setInitialTemplateMode(String mode)
```

Definiert den Namen des Anfangsvorlagenmodus. Bei Beginn der Verarbeitung erhalten die Vorlagen diesen Moduswert. Die Transformation muss nach Zuweisung des XML- und XSLT-Dokuments gestartet werden.

Parameter:

`mode`: Der Name des Anfangsvorlagenmodus in Form eines String.

Nach oben | Methoden | Enumerationen

setInputXMLFileName

```
public void setInputXMLFileName(String xmlFile)
```

Definiert den Pfad des zu transformierenden XML-Dokuments in Form einer URL.

Parameter:

`xmlFile`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden XML-Datei angibt.

Nach oben | Methoden | Enumerationen

setInputXMLFromText

```
public void setInputXMLFromText(String xmlText)
```

Liefert den Inhalt des XML-Input-Dokuments als Text.

Parameter:

`xmlText`: Beim bereitgestellten String handelt es sich um die zu verarbeitenden XML-Daten.

Nach oben | Methoden | Enumerationen

setLoadXMLWithPSVI

```
public void setLoadXMLWithPSVI(boolean load)
```

Aktiviert bzw. deaktiviert die Option zum Laden und Verwenden des Post Schema Validation Infoset (PSVI). Wenn das PSVI geladen wird, können Daten im XML-Dokument anhand der Informationen aus dem Schema qualifiziert werden. Der Wert `true` aktiviert das Laden des PSVI; `false` deaktiviert es.

Parameter:

`load`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

setNamedTemplateEntryPoint

```
public void setNamedTemplateEntryPoint(String template)
```

Liefert den Namen der benannten Vorlage, mit der die Verarbeitung beginnen soll.

Parameter:

`template`: Der Name der benannten Vorlage in Form eines String.

Nach oben | Methoden | Enumerationen

setSchemaImports

```
public void setSchemaImports(ENUMSchemaImports opt)
```

Definiert, wie Schemaimporte auf Basis der Attributwerte der `xs:import` Elemente behandelt werden sollen. Die Art der Behandlung wird durch das ausgewählte `ENUMSchemaImports` definiert.
Parameter:
`opt`: Enthält das `ENUMSchemaImports` Literal, welches festlegt, wie Schemaimporte zu behandeln sind.

Nach oben | Methoden | Enumerationen

setSchemalocationHints

```
public void setSchemalocationHints(ENUMLoadSchemalocation opt)
```

Definiert, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll. Der Mechanismus wird durch das ausgewählte `ENUMLoadSchemalocation` Literal definiert.
Parameter:
`opt`: Enthält das `ENUMLoadSchemalocation` Literal, welches festlegt, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll.

Nach oben | Methoden | Enumerationen

setSchemaMapping

```
public void setSchemaMapping(ENUMSchemaMapping opt)
```

Definiert, welches Mapping zum Auffinden des Schemas verwendet werden soll. Das Mapping wird durch das ausgewählte `ENUMSchemaMapping` definiert.
Parameter:
`opt`: Enthält das `ENUMSchemaMapping` Literal.

Nach oben | Methoden | Enumerationen

setStreamingSerialization

```
public void setStreamingSerialization(boolean support)
```

Aktiviert die Streaming-Serialisierung. Im Streaming-Modus werden möglichst wenige Daten im Arbeitsspeicher behalten, wodurch die Verarbeitung beschleunigt wird.
Parameter:
`support`: Der Wert `true` aktiviert die Streaming-Serialisierung; `false` deaktiviert sie.

Nach oben | Methoden | Enumerationen

setVersion

```
public void setVersion(EnumXSLTVersion version)
```

Definiert die für die Verarbeitung (Validierung oder XSLT-Transformation) zu verwendende XSLT-Version.
Parameter:
`version`: Enthält ein `EnumXSLTVersion` Enumerationsliteral `eVersion10`, `eVersion20` oder `eVersion30`.

 © 2014 Altova GmbH

Nach oben | Methoden | Enumerationen

setXincludeSupport

`public void setXIncludeSupport(boolean support)`

Aktiviert bzw. deaktiviert die Verwendung von `XInclude` Elementen. Der Wert `true` aktiviert die XInclude-Unterstützung; `false` deaktiviert sie. Der Standardwert ist `false`.

Parameter:

`support`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

setXMLValidationMode

`public void setXMLValidationMode(ENUMXMLValidationMode mode)`

Definiert den XML-Validierungsmodus, welcher ein Enumerationsliteral von `ENUMXMLValidationMode` ist.

Parameter:

`mode`: Ist ein Enumerationsliteral von `ENUMXMLValidationMode`.

Nach oben | Methoden | Enumerationen

setXSDVersion

`public void setXSDVersion(ENUMXSDVersion version)`

Definierrt die XML-Schema-Version, anhand welcher das XML-Dokument validiert wird.

Parameter:

`version`: Ist ein Enumerationsliteral von `ENUMXSDVersion`.

Nach oben | Methoden | Enumerationen

setXSLFileName

`public void setXSLFileName(String xslFile)`

Definiert den Pfad des für die Transformation zu verwendenden XSLT-Dokuments in Form einer URL.

Parameter:

`xslFile`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der XSLT-Datei angibt.

Nach oben | Methoden | Enumerationen

setXSLFromText

```
public void setXSLFromText(String xslText)
```
Liefert den Inhalt des XSLT-Dokuments als Text.
Parameter:
`xslText`: Der bereitgestellte String ist das für die Transformation zu verwendende XSLT-Dokument.

Nach oben | Methoden | Enumerationen

Enumerationen
ENUMXSLTVersion

ENUMXSLTVersion

```
public enum ENUMXSLTVersion {
       eVersion10
       eVersion20
       eVersion30 }
```

`ENUMXSLTVersion` erhält eines der Enumerationsliterale: `eVersion10`, `eVersion20`, `eVersion30`. Diese definieren die für die Verarbeitung (Validierung oder XSLT-Transformation) zu verwendende XSLT-Version.

Verwendet von (Schnittstelle::Methode):

XSLT	setVersion

Nach oben | Methoden | Enumerationen

6.2.4 XQuery

```
public interface XQuery
```

Beschreibung
Führt XQuery 1.0- und 3.0-Dokumente mit dem RaptorXML-Prozessor aus. XQuery- und XML-Dokumente können als Datei (über eine URL) oder als Textstring bereitgestellt werden. Es können externe XQuery-Variablen bereitgestellt werden und es steht eine Reihe von Serialisierungsoptionen zur Verfügung. Das XQuery-Dokument kann auch validiert werden. Wo Strings als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden.

Zuerst sind die Methoden der Schnittstelle und anschließend ihre Enumerationen beschrieben.

Methoden
Die Methoden der Klasse sind nachstehend in alphabetischer Reihenfolge beschrieben. In der Tabelle sind sie aus Gründen der Übersichtlichkeit nach Gruppen geordnet.

Verarbeitung	XML	XQuery
isValid	setInputXMLFileName	setVersion
isValidUpdate	setInputXMLFromText	setXQueryFileName
execute	setLoadXMLWithPSVI	setXQueryFromText
executeAndGetResultAsString	setXincludeSupport	addExternalVariable
executeUpdate	setXMLValidationMode	clearExternalVariableList
executeUpdateAndGetResultAsString	setXSDVersion	
getLastErrorMessage		
setUpdatedXMLWriteMode		

Serialisierungsoptionen	Erweiterungen
setIndentCharacters	setChartExtensionsEnabled
setKeepFormatting	setDotNetExtensionsEnabled
setOutputEncoding	setJavaExtensionsEnabled
setOutputIndent	
setOutputMethod	
setOutputOmitXMLDeclaration	

Nach oben | Methoden | Enumerationen

addExternalVariable

```
public void addExternalVariable(String name, String value)
```
Fügt den Namen und Wert einer neuen externen Variablen hinzu. Jede externe Variable und ihr Wert müssen in einem separaten Methodenaufruf definiert werden. Variablen müssen im XQuery-Dokument (mit einer optionalen Typdeklaration) deklariert werden. Setzen Sie den Variablenwert in einfache Anführungszeichen, wenn der Variablenwert ein String ist.

Parameter:
`name`: Enthält den Namen der Variablen, welcher ein QName in Form eines String ist.
`value`: Enthält den Wert der Variablen als String.

Nach oben | Methoden | Enumerationen

clearExternalVariableList

```
public void clearExternalVariableList()
```
Löscht die mit der Methode `AddExternalVariable` erstellte Liste der externen Variablen.

Nach oben | Methoden | Enumerationen

execute

```
public boolean execute(String outputFile)
```
Führt die XQuery-Transformation anhand der in `ENUMXSLTVersion` (siehe `setVersion` Methode) genannten XQuery-Spezifikation aus und speichert das Ergebnis in der im Parameter `outputFile` genannten Ausgabedatei.
Parameter:
`outputFile`: Ein String, der den Pfad (Pfad und Dateinamen) der Ausgabedatei liefert.
Rückgabe:
Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`.

Nach oben | Methoden | Enumerationen

executeUpdate

```
public boolean executeUpdate(String outputFile)
```
Führt das XQuery Update entsprechend der in `ENUMXQueryVersion` (siehe `setVersion` Methode) genannten XQuery Update-Spezifikation aus und speichert das Ergebnis in der im Parameter `outputFile` angegebenen Ausgabedatei.
Parameter:
`outputFile`: Ein String, der den Pfad (Pfad und Dateinamen) der Ausgabedatei angibt.
Rückgabe:
Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`.

Nach oben | Methoden | Enumerationen

executeAndGetResultAsString

```
public String executeAndGetResultAsString()
```
Führt die XQuery-Transformation anhand der in `ENUMXSLTVersion` (siehe `setVersion` Methode)

genannten XQuery-Spezifikation aus und gibt das Ergebnis als String zurück.
Rückgabe:
ein String, der das Ergebnis der XQuery-Ausführung ist.

Nach oben | Methoden | Enumerationen

getLastErrorMessage
`public String` **`getLastErrorMessage()`**
Ruft die letzte Fehlermeldung vom XQuery-Prozessor auf.
Rückgabe:
ein String, der die letzte Fehlermeldung vom XQuery-Prozessor ist.

Nach oben | Methoden | Enumerationen

isValid
`public boolean` **`isValid()`**
Gibt das Ergebnis der Validierung des XQuery-Dokuments, die anhand der in `ENUMXQueryVersion` (siehe `setVersion` Methode) genannten XQuery-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `getLastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.
Rückgabe:
Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`.

Nach oben | Methoden | Enumerationen

isValidUpdate
`public boolean` **`isValidUpdate()`**
Gibt das Ergebnis der Validierung des XQuery Update-Dokuments, das entsprechend der in `ENUMXQueryVersion` (siehe `setVersion` Methode) genannten XQuery Update-Spezifikation ausgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `getLastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.
Rückgabe:
Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`.

Nach oben | Methoden | Enumerationen

setChartExtensionsEnabled
`public void` **`setChartExtensionsEnabled(`**`boolean enable`**`)`**

Aktiviert bzw. deaktiviert die Diagrammerweiterungsfunktionen von Altova.
Parameter:
`enable`: Der Wert `true` aktiviert Diagrammerweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

Nach oben | Methoden | Enumerationen

setDotNetExtensionsEnabled
`public void setDotNetExtensionsEnabled(boolean enable)`
Aktiviert oder deaktiviert .NET-Erweiterungsfunktionen.
Parameter:
`enable`: Der Wert `true` aktiviert .NET-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

Nach oben | Methoden | Enumerationen

setIndentCharacters
`public void setIndentCharacters(String chars)`
Definiert den in der Ausgabe als Einrückung zu verwendenden Zeichenstring.
Parameter:
`chars`: Enthält den Einrückungszeichenstring.

Nach oben | Methoden | Enumerationen

setInputXMLFileName
`public void setInputXMLFileName(String xmlFile)`
Definiert den Pfad des für die XQuery-Ausführung zu verwendenden XML-Dokuments in Form einer URL.
Parameter:
`xmlFile`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden XML-Datei angibt.

Nach oben | Methoden | Enumerationen

setInputXMLFromText
`public void setInputXMLFromText(String xmlText)`
Liefert den Inhalt des XML-Input-Dokuments als Text.
Parameter:
`xmlText`: Beim bereitgestellten String handelt es sich um die zu verarbeitenden XML-Daten.

 © 2014 Altova GmbH

Nach oben | Methoden | Enumerationen

setJavaExtensionsEnabled

```
public void setJavaExtensionsEnabled(boolean enable)
```

Aktiviert bzw. deaktiviert Java-Erweiterungsfunktionen.

Parameter:

`enable`: Der Wert `true` aktiviert Java-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

Nach oben | Methoden | Enumerationen

setKeepFormatting

```
public void setKeepFormatting(boolean keep)
```

Aktiviert bzw. deaktiviert die Option zur Beibehaltung der Originalformatierung von Dateien, die mit `executeUpdate` aktualisiert werden.

Parameter:

`keep`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

setLoadXMLWithPSVI

```
public void setLoadXMLWithPSVI(boolean load)
```

Aktiviert bzw. deaktiviert die Option zum Laden und Verwenden des Post Schema Validation Infoset (PSVI). Wenn das PSVI geladen wird, können Daten im XML-Dokument anhand der Informationen aus dem Schema qualifiziert werden. Der Wert `true` aktiviert das Laden des PSVI; `false` deaktiviert es.

Parameter:

`load`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

setOutputEncoding

```
public void setOutputEncoding(String encoding)
```

Definiert die Kodierung des Ergebnisdokuments.

Parameter:

`encoding`: Verwenden Sie den Namen einer offiziellen IANA-Kodierung wie z.B. `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1` als String.

Nach oben | Methoden | Enumerationen

setOutputIndent

```
public void setOutputIndent(boolean indent)
```
Aktiviert bzw. deaktiviert die Einrückung im Ausgabedokument.
Parameter:
`indent`: Der Wert `true` aktiviert die Einrückung; `false` deaktiviert sie.

Nach oben | Methoden | Enumerationen

setOutputMethod

```
public void setOutputMethod(String outputMethod)
```
Definiert die Serialisierung des Ausgabedokuments.
Parameter:
`outputMethod`: Gültige Werte sind: `xml | xhtml | html | text` in Form eines String. Der Standardwert ist `xml`.

Nach oben | Methoden | Enumerationen

setOutputOmitXMLDeclaration

```
public void setOutputOmitXMLDeclaration(boolean omit)
```
Aktiviert bzw. deaktiviert den Einschluss der XML-Deklaration in das Ergebnisdokument.
Parameter:
`omit`: Der Wert `true` lässt die Deklaration weg; `false` inkludiert sie. Der Standardwert ist `false`.

Nach oben | Methoden | Enumerationen

setUpdatedXMLWriteMode

```
public void setUpdatedXMLWriteMode(EnumXQueryUpdatedXML mode)
```
Definiert den Modus, der für die Aktualisierung verwendet werden soll.
Parameter:
Enthält ein `EnumXQueryUpdatedXML` Enumerationsliteral `eUpdatedDiscard`, `eUpdatedWriteback` oder `eUpdatedAsMainResult`.

Nach oben | Methoden | Enumerationen

setVersion

```
public void setVersion(EnumXQueryVersion version)
```
Definiert die für die Verarbeitung (Validierung oder XQuery-Ausführung) zu verwendende XQuery-Version.
Parameter:
`version`: Enthält ein `EnumXQueryVersion` Enumerationsliteral `eVersion10` oder `eVersion30`. Der Standardwert ist `eVersion30ml`.

Nach oben | Methoden | Enumerationen

setXincludeSupport

```
public void setXIncludeSupport(boolean support)
```

Aktiviert bzw. deaktiviert die Verwendung von `XInclude` Elementen. Der Wert `true` aktiviert die XInclude-Unterstützung; `false` deaktiviert sie. Der Standardwert ist `false`.

Parameter:

`support`: Erhält den Booleschen Wert `true` oder `false`.

Nach oben | Methoden | Enumerationen

setXMLValidationMode

```
public void setXMLValidationMode(ENUMXMLValidationMode mode)
```

Definiert den XML-Validierungsmodus, welcher ein Enumerationsliteral von `ENUMXMLValidationMode` ist.

Parameter:

`mode`: Ist ein Enumerationsliteral von `ENUMXMLValidationMode`.

Nach oben | Methoden | Enumerationen

setXQueryFileName

```
public void setXQueryFileName(String queryFile)
```

Definiert den Pfad der auszuführenden XQuery-Datei in Form einer URL.

Parameter:

`queryFile`: Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden XML-Datei angibt.

Nach oben | Methoden | Enumerationen

setXQueryFromText

```
public void setXQueryFromText(String queryText)
```

Liefert den Inhalt des XQuery-Dokuments als Text.

Parameter:

`queryText`: Der bereitgestellte String ist das zu verarbeitende XQuery-Dokument.

Nach oben | Methoden | Enumerationen

setXSDVersion

```
public void setXSDVersion(ENUMXSDVersion version)
```

Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert werden soll.
Parameter:
`version`: Ist ein Enumerationsliteral von `ENUMXSDVersion`.

Nach oben | Methoden | Enumerationen

Enumerationen
ENUMXQueryUpdatedXML
ENUMXQueryVersion

ENUMXQueryUpdatedXML

```
public enum ENUMXQueryUpdatedXML {
       eUpdatedDiscard
       eUpdatedWriteback
       eeUpdatedAsMainResult }
```

`ENUMXQueryVersion` erhält eines der folgenden Enumerationsliterale:

- `eUpdatedDiscard`: Aktualisierungen werden verworfen und nicht in eine Datei geschrieben.
- `eUpdatedWriteback`: Aktualisierungen werden in die mit `setInputXMLFileName` definierte XML-Input-Datei geschrieben.
- `eUpdatedAsMainResult`: Aktualisierungen werden unter dem mit dem Parameter `outputFile` von `ExecuteUpdate` definierten Pfad gespeichert.

Verwendet von (Schnittstelle::Methode):
`XQuery` `setUpdatedXMLWriteMode`

Nach oben | Methoden | Enumerationen

ENUMXQueryVersion

```
public enum ENUMXQueryVersion {
       eVersion10
       eVersion30 }
```

`ENUMXQueryVersion` erhält eines der Enumerationsliterale: `eVersion10`, `eVersion30`. Diese definieren, welche XQuery-Version für die Verarbeitung (Ausführung oder Validierung) verwendet werden soll.

Verwendet von (Schnittstelle::Methode):
`XQuery` `setVersion`

Nach oben | Methoden | Enumerationen

 © 2014 Altova GmbH

6.2.5 RaptorXMLException

```
public interface RaptorXMLException
```

Beschreibung
Hat eine einzige Methode, die die Ausnahme generiert.

RaptorXMLException
```
public void RaptorXMLException(String message)
```
Generiert eine Ausnahme, die Informationen zum Fehler, der bei der Verarbeitung aufgetreten ist, enthält.
Parameter:
`message`: Ein String mit Informationen über den Fehler.

© 2014 Altova GmbH

Kapitel 7

COM- und .NET-Schnittstelle

7 COM- und .NET-Schnittstelle

Zwei Schnittstellen, eine API
Die COM- und die .NET-Schnittstelle von RaptorXML Server verwenden eine einzige API: die COM/.NET API von RaptorXML Server. Die .NET-Schnittstelle bildet einen Wrapper rund um die COM-Schnittstelle.

Sie können RaptorXML mit folgenden Sprachen verwenden:

- Skriptsprachen wie JavaScript, über die COM-Schnittstelle
- Programmiersprachen wie C#, über die .NET-Schnittstelle

Gliederung dieses Abschnitts
Dieser Abschnitt ist folgendermaßen gegliedert:

- Informationen zur COM-Schnittstelle: Hier wird beschrieben, wie die COM-Schnittstelle funktioniert und welche Schritte nötig sind, um die COM-Schnittstelle zu nutzen.
- Informationen zur .NET-Schnittstelle: Hier wird beschrieben, wie Sie Ihre Umgebung für die Verwendung der .NET-Schnittstelle einrichten.
- Programmiersprachen: Hier finden Sie Codefragmente in gebräuchlichen Programmiersprachen, in denen gezeigt wird, wie RaptorXML-Funktionalitäten aufgerufen werden.
- Die API-Referenz: Hier sind das Objektmodell, die Objekte und Eigenschaften der API dokumentiert.

© 2014 Altova GmbH

7.1 Informationen zur COM-Schnittstelle

RaptorXML Server wird bei der Installation von RaptorXML Server automatisch als COM-Serverobjekt registriert, sodass das Programm von Applikationen und Skriptsprachen mit Unterstützung für COM-Aufrufe aufgerufen werden kann. Wenn Sie den Pfad des RaptorXML Server Installationspakets ändern möchten, sollten Sie RaptorXML Server am besten zuerst deinstallieren und anschließend im gewünschten Ordner installieren. Auf diese Art wird die Aufhebung der Registrierung und die erneute Registrierung automatisch während der Installation vorgenommen.

Überprüfen, ob die Registrierung erfolgreich vorgenommen wurde
Wenn die Registrierung erfolgreich war, enthält die Registrierungsdatei die Klassen `RaptorXML.Server`. Diese beiden Klassen befinden sich normalerweise unter `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`.

Codebeispiel
Im Abschnitt Programmiersprachen finden Sie ein VBScript-Beispiel, in dem gezeigt wird, wie die RaptorXML API über ihre COM-Schnittstelle verwendet werden kann. Eine Beispieldatei dazu finden Sie im RaptorXML Applikationsordner im Ordner `examples/API`.

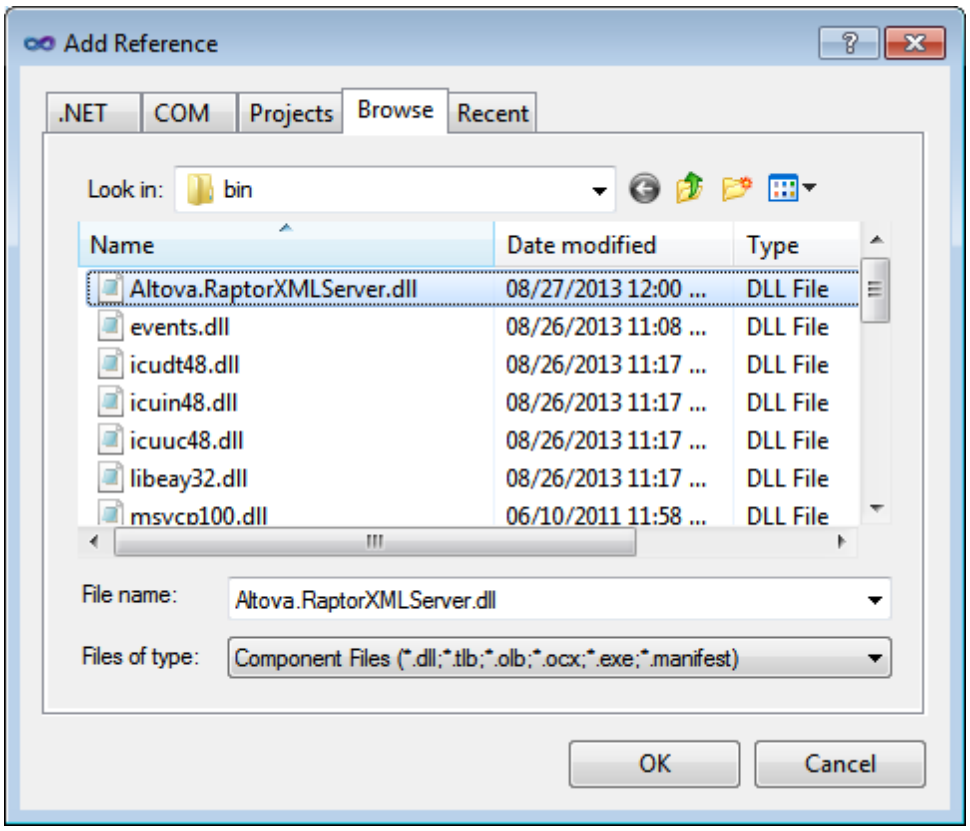
7.2 Informationen zur .NET-Schnittstelle

Die .NET-Schnittstelle ist als Wrapper rund um die RaptorXML COM-Schnittstelle gebaut. Sie dient als primäre von Altova signierte Interop-Assembly und verwendet den Namespace `Altova.RaptorXMLServer`.

Hinzufügen der RaptorXML DLL als Referenz zum Visual Studio .NET-Projekt
Um RaptorXML in Ihrem .NET-Projekt verwenden zu können, fügen Sie in Ihrem Projekt eine Referenz zur RaptorXML-DLL (`Altova.RaptorXMLServer.dll`) hinzu. Ihre RaptorXML Server Installation enthält eine signierte DLL-Datei namens `Altova.RaptorXMLServer.dll`, die bei der Installation von RaptorXML mit Hilfe des RaptorXML Installationsprogramms automatisch zum globalen Assembly Cache (GAC) hinzugefügt wird. Normalerweise befindet sich der GAC im Ordner `C:\WINDOWS\assembly`.

Um die RaptorXML-DLL als Referenz in einem .NET-Projekt hinzuzufügen, gehen Sie folgendermaßen vor:

1. Klicken Sie bei geöffnetem .NET-Projekt auf **Project | Add Reference**. Daraufhin wird das Dialogfeld "Add Reference" (*Abbildung unten*) angezeigt



1. Navigieren Sie auf dem Register "Browse" zum Ordner: `<RaptorXML application folder>/bin`, wählen Sie die RaptorXML DLL `Altova.RaptorXMLServer.dll` aus und klicken Sie auf **OK**.

2. Wählen Sie den Befehl View | Object Browser, um die Objekte der RaptorXML API zu sehen.

Sobald die `Altova.RaptorXMLServer.dll` der .NET-Schnittstelle zur Verfügung steht und RaptorXML als COM-Serverobjekt registriert wurde, stehen die RaptorXML-Funktionalitäten in Ihrem .NET-Projekt zur Verfügung.

Anmerkung: RaptorXML wird bei der Installation automatisch als COM-Serverobjekt registriert, daher ist eine manuelle Registrierung nicht erforderlich.

Anmerkung: Falls Sie einen Zugriffsfehler erhalten, vergewissern Sie sich, dass die Berechtigungen richtig eingestellt sind. Gehen Sie zu "Component Services" und geben Sie demselben Benutzerkonto, über das der Application Pool, der RaptorXML enthält, ausgeführt wird, Zugriffsberechtigungen.

Codebeispiele
Im Abschnitt Programmiersprachen finden Sie ein C#-Beispiel und ein Visual Basic .NET-Beispiel zur Verwendung der RaptorXML API über ihre .NET-Schnittstelle. Die Dateien zu diesen Beispielen befinden sich im RaptorXML-Applikationsordner im Ordner `examples/API`.

7.3 Programmiersprachen

Programmiersprachen unterscheiden sich in der Art, wie sie den COM- / .NET-Zugriff unterstützen. Zur Einführung erhalten Sie einige Beispiele für die gebräuchlichsten Sprachen (*Links siehe unten*). In den Codefragmenten in diesem Abschnitt wird gezeigt, wie Sie grundlegende Funktionalitäten aufrufen. Sie finden diese grundlegenden Funktionalitäten im RaptorXML Server Applikationsordner im Ordner `examples/API`.

VBScript
Sie können über VBScript auf die COM API von RaptorXML Server zugreifen. Anhand des VBScript Codefragments werden die folgenden Funktionalitäten gezeigt:

- Verbinden mit der RaptorXML Server COM API
- Validieren einer XML-Datei
- Ausführen einer XSL-Transformation
- Durchführen einer XQuery-Ausführung

C#
Sie können über C# auf die COM API von RaptorXML Server zugreifen. Anhand des C#-Codefragments werden die folgenden Funktionalitäten gezeigt:

- Verbinden mit der RaptorXML Server .NET API
- Validieren einer XML-Datei
- Ausführen einer XSL-Transformation
- Durchführen einer XQuery-Ausführung

Visual Basic .NET
Visual Basic.NET unterscheidet sich nur in der Syntax von C#. Die .NET API wird auf die gleiche Art wie bei C# aufgerufen. Anhand des Visual Basic-Codefragments werden die folgenden Funktionalitäten gezeigt:

- Verbinden mit der RaptorXML Server .NET API
- Validieren einer XML-Datei
- Ausführen einer XSL-Transformation
- Durchführen einer XQuery-Ausführung

Dieser Abschnitt enthält die folgenden Codebeispiele:

Für die COM-Schnittstelle

- Ein Beispiel in VBScript

Für die .NET-Schnittstelle

- Ein Beispiel in C#
- Ein Beispiel in Visual Basic

 © 2014 Altova GmbH

7.3.1 COM-Beispiel: VBScript

Das VBScript-Beispiel ist in die folgenden Abschnitte gegliedert:

- Einrichten und Initialisieren des RaptorXML COM-Objekts
- Validieren einer XML-Datei
- Durchführung einer XSLT-Transformation, Rückgabe des Ergebnisses als String
- Verarbeiten eines XQuery-Dokuments, Speichern des Ergebnisses in einer Datei
- Einrichten der Ausführungssequenz des Skripts und seines Eintrittspunkts

---

```
' The RaptorXML COM object
dim objRaptor

' Initialize the RaptorXML COM object
sub Init
	objRaptor = Null
	On Error Resume Next
	' Try to load the 32-bit COM object; do not throw exceptions if object is
not found
	Set objRaptor = WScript.GetObject( "", "RaptorXML.Server" )
	On Error Goto 0
	if ( IsNull( objRaptor ) ) then
		' Try to load the 64-bit object (exception will be thrown if not
found)
		Set objRaptor = WScript.GetObject( "", "RaptorXML_x64.Server" )
	end if
	' Configure the server: error reporting, HTTP server name and port (IPv6
localhost in this example)
	objRaptor.ErrorLimit = 1
	objRaptor.ReportOptionalWarnings = true
	objRaptor.ServerName = "::1"
	objRaptor.ServerPort = 8087
end sub
```

```
' Validate one file
sub ValidateXML
	' Get a validator instance from the Server object
	dim objXMLValidator
	Set objXMLValidator = objRaptor.GetXMLValidator()

	' Configure input data
	objXMLValidator.InputXMLFileName = "MyXMLFile.xml"

	' Validate; in case of invalid file report the problem returned by
RaptorXML
	if ( objXMLValidator.IsValid() ) then
		MsgBox( "Input string is valid" )
	else
		MsgBox( objXMLValidator.LastErrorMessage )
```

```
	end if
end sub
```

```
' Perform a transformation; return the result as a string
sub RunXSLT
	' Get an XSLT engine instance from the Server object
	dim objXSLT
	set objXSLT = objRaptor.GetXSLT

	' Configure input data
	objXSLT.InputXMLFileName = "MyXMLFile.xml"
	objXSLT.XSLFileName = "MyTransformation.xsl"

	' Run the transformation; in case of success the result will be returned,
in case of errors the engine returns an error listing
	MsgBox( objXSLT.ExecuteAndGetResultAsString() )
end sub
```

```
' Execute an XQuery; save the result in a file
sub RunXQuery
	' Get an XQuery engine instance from the Server object
	dim objXQ
	set objXQ = objRaptor.GetXQuery()

	' Configure input data
	objXQ.InputXMLFileName = "MyXMLFile.xml"
	objXQ.XQueryFileName = "MyQuery.xq"

	' Configure serialization (optional - for fine-tuning the result's
formatting)
	objXQ.OutputEncoding = "UTF8"
	objXQ.OutputIndent = true
	objXQ.OutputMethod = "xml"
	objXQ.OutputOmitXMLDeclaration = false

	' Run the query; the result will be serialized to the given path
	call objXQ.Execute( "MyQueryResult.xml" )
end sub
```

```
' Perform all sample functions
sub main
	Init
	ValidateXML
	RunXSLT
	RunXQuery
end sub

' Script entry point; run the main function
main
```

 © 2014 Altova GmbH

7.3.2 .NET-Beispiel: C#

Das C#-Beispiel ist in die folgenden Abschnitte gegliedert:

- Einrichten und Initialisieren des RaptorXML .NET-Objekts
- Validieren einer XML-Datei
- Durchführung einer XSLT-Transformation, Rückgabe des Ergebnisses als String
- Verarbeiten eines XQuery-Dokuments, Speichern des Ergebnisses in einer Datei
- Einrichten der Ausführungssequenz des Skripts und seines Eintrittspunkts

---

```
using System;
using System.Text;
using Altova.RaptorXMLServer;

namespace RaptorXMLRunner
{
    class Program
    {
    // The RaptorXML Server .NET object
       static ServerClass objRaptorXMLServer;

    // Initialize the RaptorXML Server .NET object
       static void Init()
       {
       // Allocate a RaptorXML Server object
          objRaptorXMLServer = new ServerClass();

       // Configure the server: error reporting, HTTP server name and port
       // (IPv6 localhost in this example)
          objRaptorXMLServer.ErrorLimit = 1;
          objRaptorXMLServer.ReportOptionalWarnings = true;
          objRaptorXMLServer.ServerName = "::1"
          objRaptorXMLServer.ServerPort = 8087
       }
```

```
       // Validate one file
          static void ValidateXML()
          {
          // Get a validator engine instance from the Server object
             XMLValidator objXMLValidator =
objRaptorXMLServer.GetXMLValidator();

          // Configure input data
             objXMLValidator.InputXMLFileName = "MyXMLFile.xml";

          // Validate; in case of invalid file,
             report the problem returned by RaptorXML
```

```
        if ( objXMLValidator.IsValid() )
           Console.WriteLine( "Input string is valid" );
        else
           Console.WriteLine( objXMLValidator.LastErrorMessage );
     }
```

```
// Perform an XSLT transformation, and
// return the result as a string
   static void RunXSLT()
   {
   // Get an XSLT engine instance from the Server object
      XSLT objXSLT = objRaptorXMLServer.GetXSLT();

      // Configure input data
         objXSLT.InputXMLFileName = "MyXMLFile.xml";
         objXSLT.XSLFileName = "MyTransformation.xsl";

     // Run the transformation.
     // In case of success, the result is returned.
     // In case of errors, an error listing
        Console.WriteLine( objXSLT.ExecuteAndGetResultAsString() );
   }
```

```
// Execute an XQuery, save the result in a file
   static void RunXQuery()
   {
   // Get an XQuery engine instance from the Server object
      XQuery objXQuery = objRaptorXMLServer.GetXQuery();

     // Configure input data
        objXQuery.InputXMLFileName = exampleFolder + "simple.xml";
        objXQuery.XQueryFileName = exampleFolder + "CopyInput.xq";

    // Configure serialization (optional, for better formatting)
       objXQuery.OutputEncoding = "UTF8"
       objXQuery.OutputIndent = true
       objXQuery.OutputMethod = "xml"
       objXQuery.OutputOmitXMLDeclaration = false

    // Run the query; result serialized to given path
       objXQuery.Execute( "MyQueryResult.xml" );
   }
```

```
   static void Main(string[] args)
```

```
            {
              try
              {
              // Entry point. Perform all functions
                 Init();
                 ValidateXML();
                 RunXSLT();
                 RunXQuery();
              }
              catch (System.Exception ex)
              {
                  Console.WriteLine( ex.Message );
                  Console.WriteLine( ex.ToString() );
              }
          }
      }
  }
```


© 2014 Altova GmbH

7.3.3 .NET-Beispiel: Visual Basic .NET

Das VBScript-Beispiel ist in die folgenden Abschnitte gegliedert:

- Einrichten und Initialisieren des RaptorXML .NET-Objekts
- Validieren einer XML-Datei
- Durchführung einer XSLT-Transformation, Rückgabe des Ergebnisses als String
- Verarbeiten eines XQuery-Dokuments, Speichern des Ergebnisses in einer Datei
- Einrichten der Ausführungssequenz des Skripts und seines Eintrittspunkts

---

```
Option Explicit On
Imports Altova.RaptorXMLServer

Module RaptorXMLRunner

   ' The RaptorXML .NET object
     Dim objRaptor As Server

   ' Initialize the RaptorXML .NET object
     Sub Init()

       ' Allocate a RaptorXML object
         objRaptor = New Server()

       ' Configure the server: error reporting, HTTP server name and port (IPv6
localhost in this example)
         objRaptor.ErrorLimit = 1
         objRaptor.ReportOptionalWarnings = True
         objRaptor.ServerName = "::1"
         objRaptor.ServerPort = 8087
     End Sub
```

```
   ' Validate one file
     Sub ValidateXML()

       ' Get a validator instance from the RaptorXML object
         Dim objXMLValidator As XMLValidator
         objXMLValidator = objRaptor.GetXMLValidator()

       ' Configure input data
         objXMLValidator.InputXMLFileName = "MyXMLFile.xml"

       ' Validate; in case of invalid file report the problem returned by
RaptorXML
         If (objXMLValidator.IsValid()) Then
            Console.WriteLine("Input string is valid")
         Else
            Console.WriteLine(objXMLValidator.LastErrorMessage)
         End If
     End Sub
```

```
    ' Perform a transformation; return the result as a string
      Sub RunXSLT()

       ' Get an XSLT engine instance from the Server object
         Dim objXSLT As XSLT
         objXSLT = objRaptor.GetXSLT()

       ' Configure input data
         objXSLT.InputXMLFileName = "MyXMLFile.xml"
         objXSLT.XSLFileName = "MyTransformation.xsl"

       ' Run the transformation; in case of success the result will be returned,
in case of errors the engine returns an error listing
         Console.WriteLine(objXSLT.ExecuteAndGetResultAsString())
      End Sub
```

```
    ' Execute an XQuery; save the result in a file
      Sub RunXQuery()

       ' Get an XQuery engine instance from the Server object
         Dim objXQ As XQuery
         objXQ = objRaptor.GetXQuery()

       ' Configure input data
         objXQ.InputXMLFileName = "MyXMLFile.xml"
         objXQ.XQueryFileName = "MyQuery.xq"

       ' Configure serialization (optional - for fine-tuning the result's
formatting)
         objXQ.OutputEncoding = "UTF8"
         objXQ.OutputIndent = true
         objXQ.OutputMethod = "xml"
         objXQ.OutputOmitXMLDeclaration = false

       ' Run the query; the result will be serialized to the given path
         objXQ.Execute( "MyQueryResult.xml" )
      End Sub
```

```
      Sub Main()
       ' Entry point; perform all sample functions
         Init()
         ValidateXML()
         RunXSLT()
         RunXQuery()
      End Sub

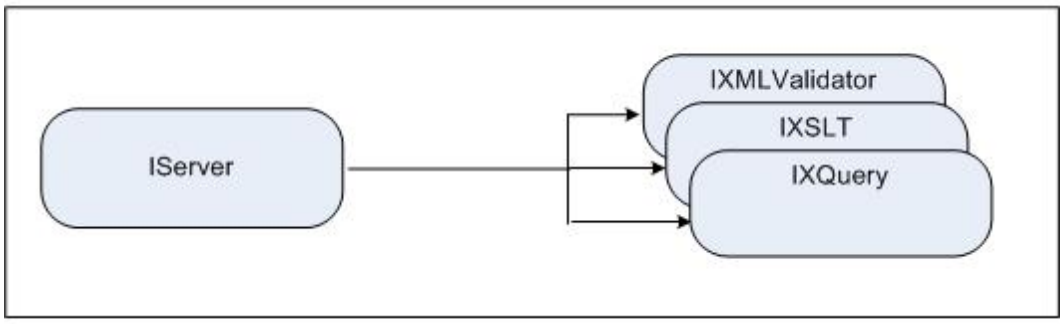
End Module
```

 © 2014 Altova GmbH

7.4 API-Referenz

In diesem Abschnitt wird die API-Spezifikation beschrieben: ihr Objektmodell und die Einzelheiten zu ihrer Schnittstelle und ihren Enumerationen.

Der Ausgangspunkt für die Verwendung der Funktionalitäten von RaptorXML ist die `IServer`-Schnittstelle. Dieses Objekt enthält die Objekte, die die RaptorXML-Funktionalitäten bereitstellen: XML-Validierung, XSLT-Transformationen und die Verarbeitung von XQuery-Dokumenten. Das Objektmodell der RaptorXML API ist im folgenden Diagramm dargestellt.



Unten sehen Sie die Hierarchie des Objektmodells. Im dazugehörigen Abschnitt werden die jeweiligen Schnittstellen beschrieben. Die Methoden und Eigenschaften der einzelnen Schnittstellen werden im Abschnitt zur jeweiligen Schnittstelle beschrieben.

```
-- IServer
  |-- IXMLValidator
  |-- IXSLT
  |-- IXQuery
```

7.4.1 Schnittstellen

Es sind die folgenden Schnittstellen definiert. Sie sind in den Unterabschnitten dieses Abschnitts beschrieben.

IServer

IXMLValidator

IXSLT

IXQuery

IServer

Die **IServer** Schnittstelle liefert Methoden, um Schnittstellen des entsprechenden RaptorXML-Prozessors zurückzugeben: XML Validator, XSLT und XQuery. Die Eigenschaften definieren die Parameter der Schnittstelle.

Methoden
IXMLValidator
IXSLT
IXQuery

Eigenschaften		
APIMajorVersion	GlobalResourcesFile	ServerName
APIMinorVersion	Is64Bit	ServerPath
APIServicePackVersion	MajorVersion	ServerPort
ErrorFormat	MinorVersion	ServicePackVersion
ErrorLimit	ProductName	UserCatalog
GlobalCatalog	ProductNameAndVersion	
GlobalResourceConfig	ReportOptionalWarnings	

Methoden

Die Methoden der **IServer** Schnittstelle geben Schnittstellen des jeweiligen RaptorXML-Prozessors zurück: XML-Validator, XSLT und XQuery.

IXMLValidator **GetXMLValidator()** [Nach oben | Methoden | Eigenschaften]
Gibt eine Instanz des XML-Validierungsprozessors zurück.

 © 2014 Altova GmbH

IXSLT **GetXSLT()** [Nach oben | Methoden | Eigenschaften]
Gibt eine Instanz des XSLT-Prozessors zurück.

IXQuery **GetXQuery()** [Nach oben | Methoden | Eigenschaften]
Gibt eine Instanz des XQuery-Prozessors zurück.

Eigenschaften
Die Eigenschaften der **IServer** Schnittstelle sind im Folgenden in alphabetischer Reihenfolge beschrieben. Aus Gründen der Übersichtlichkeit sind die Eigenschaften in der Tabelle in Gruppen gegliedert. Beachten Sie, dass String-Inputs, die als URLs interpretiert werden sollen, als absolute Pfade angegeben werden müssen. Bei Verwendung eines relativen Pfads sollte im Aufrufmodul ein Mechanismus zum Auflösen des relativen Pfads definiert sein.

Fehler und Warnungen	Kataloge	Globale Ressourcen	HTTP-Server
ErrorFormat	GlobalCatalog	GlobalResourceConfig	ServerName
ErrorLimit	UserCatalog	GlobalResourcesFile	ServerPath
ReportOptionalWarnings			ServerPort

Produktinformationen	
ProductName	Is64Bit
ProductNameAndVersion	APIMajorVersion
MajorVersion	APIMinorVersion
MinorVersion	APIServicePackVersion
ServicePackVersion	

int **APIMajorVersion** [Nach oben | Methoden | Eigenschaften]
Gibt die Hauptversion der API als Ganzzahl zurück. Die Hauptversion der API muss nicht unbedingt mit der Hauptversion des Produkts übereinstimmen, wenn die API mit einem anderen Server verbunden ist.

---

int **APIMinorVersion** [Nach oben | Methoden | Eigenschaften]
Gibt die Nebenversion der API als Ganzzahl zurück. Die Nebenversion der API muss nicht unbedingt mit der Nebenversion des Produkts übereinstimmen, wenn die API mit einem anderen Server verbunden ist.

`int` **APIServicePackVersion** [Nach oben | Methoden | Eigenschaften]
Gibt die Service Pack-Version der API als Ganzzahl zurück. Die Service Pack-Version der API muss nicht unbedingt mit der Service Pack-Version des Produkts übereinstimmen, wenn die API mit einem anderen Server verbunden ist.

`ENUMErrorFormat` **ErrorFormat** [Nach oben | Methoden | Eigenschaften]
Definiert das RaptorXML-Fehlerformat und ist ein `ENUMErrorFormat` Literal (`Text` | `ShortXML` | `LongXML`).

`int` **ErrorLimit** [Nach oben | Methoden | Eigenschaften]
Konfiguriert das Validierungsfehlerlimit für RaptorXML. Der Typ ist `uint`. Bei Erreichen des Fehlerlimits wird die Ausführung gestoppt. Der Standardwert ist `100`.

`string` **GlobalCatalog** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad zur Hauptkatalogdatei (Eintrittspunkt). Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad der zu verwendenden Katalogdatei angibt.

`string` **GlobalResourceConfig** [Nach oben | Methoden | Eigenschaften]
Definiert die aktive Konfiguration der zu verwendenden globalen Ressource.

`string` **GlobalResourcesFile** [Nach oben | Methoden | Eigenschaften]
Definiert die globale Ressourcendatei. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur zu verwendenden globalen Ressourcendatei angibt.

`bool` **Is64Bit** [Nach oben | Methoden | Eigenschaften]
Überprüft, ob die Applikation eine ausführbare 64-Bit-Datei ist. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `true` zurück.

`int` **MajorVersion** [Nach oben | Methoden | Eigenschaften]
Gibt die Hauptversion des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2014r2sp1(x64)` den Wert `16` zurück (den Unterschied zwischen der Hauptversion (`2014`) und dem Anfangsjahr `1998`).

`int` **MinorVersion** [Nach oben | Methoden | Eigenschaften]
Gibt die Nebenversion des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `2` zurück (den Unterschied zur Nebenversionsnummer `r2`).

 © 2014 Altova GmbH

`string` **ProductName** [Nach oben | Methoden | Eigenschaften]
Gibt den Namen des Produkts als String zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `Altova RaptorXML Server` zurück.

`string` **ProductNameAndVersion** [Nach oben | Methoden | Eigenschaften]
Gibt den Namen des Produkts und seine Version als Ganzzahl zurück. *Beispiel:* Gibt für `Altova RaptorXML Server 2015r2sp1(x64)` den Wert `Altova RaptorXML Server 2015r2sp1(x64)` zurück.
.

`bool` **ReportOptionalWarnings** [Nach oben | Methoden | Eigenschaften]
Aktiviert bzw. deaktiviert die Ausgabe von Warnungen. Der Wert `true` aktiviert Warnungen; `false` deaktiviert sie.

`string` **ServerName** [Nach oben | Methoden | Eigenschaften]
Definiert den Namen des HTTP-Servers. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst.

`string` **ServerPath** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad zum HTTP-Server in Form einer URL. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst.

`int` **ServerPort** [Nach oben | Methoden | Eigenschaften]
Definiert den Server-Port des HTTP-Servers. Der Typ ist `ushort`. Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst.

`int` **ServicePackVersion** [Nach oben | Methoden | Eigenschaften]
Gibt die Service Pack-Version des Produkts als Ganzzahl zurück. *Beispiel:* Gibt für `RaptorXML Server 2015r2sp1(x64)` den Wert `1` (von der Service Pack Version Nummer `sp1`) zurück.

`string` **UserCatalog** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad der benutzerdefinierten Katalogdatei in Form einer URL. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur zu verwendenden Katalogdatei angibt.

IXMLValidator

Die **`IXMLValidator`** Schnittstelle bietet Methoden zum Überprüfen folgender Eigenschaften:

- der Gültigkeit eines XML-Dokuments, einer DTD oder eines XML-Schema-Dokuments:

IsValid. XML-Dokumente können anhand einer DTD oder eines XML-Schemas validiert werden. Die Referenzen auf dieses Dokument können sich innerhalb des XML-Dokuments befinden oder über den Code geliefert werden.
- der Wohlgeformtheit eines XML-Dokuments: IsWellFormed.

Beide Methoden geben die Booleschen Werte TRUE oder FALSE zurück. Die Eigenschaften definieren die Parameter der Schnittstelle.

Methoden
IsValid
IsWellFormed

Eigenschaften		
AssessmentMode	InputXMLFromText	SchemalocationHints
DTDFileName	LastErrorMessage	SchemaMapping
DTDFromText	PythonScriptFile	SchemaTextArray
EnableNamespaces	SchemaFileArray	Streaming
InputFileArray	SchemaFileName	XincludeSupport
InputTextArray	SchemaFromText	XMLValidationMode
InputXMLFileName	SchemaImports	XSDVersion

Methoden

Die beiden Methoden der **IXMLValidator** Schnittstelle sind **IsValid** und **IsWellFormed**. Sie überprüfen die Gültigkeit bzw. Wohlgeformtheit des angegebenen Dokuments. Beide Methoden geben die Booleschen Werte true oder false zurück.

```
bool IsValid(ENUMValidationType nType)
```
[Nach oben | Methoden | Eigenschaften]

- Gibt das Ergebnis der Validierung zurück. Die Validierung wird durch den Wert von ENUMValidationType definiert. Das Ergebnis ist bei Erfolg true, bei Fehlschlag false.
- nType ist der Wert von ENUMValidationType. Der Validierungstyp definiert, ob das XML-Dokument anhand einer DTD oder einer XSD validiert werden soll und ob eine DTD oder ein XSD-Dokument validiert wird. Der Standardwert ist eValidateAny, welcher angibt, dass der Typ des Dokuments von RaptorXML automatisch ermittelt werden soll.
- Wenn ein Fehler auftritt, wird eine RaptorXMLException ausgegeben. Mit Hilfe der LastErrorMessage Methode können Sie zusätzliche Informationen aufrufen.

 © 2014 Altova GmbH

bool **IsWellFormed**(ENUMWellformedCheckType nType) [Nach oben | Methoden | Eigenschaften]

- Gibt das Ergebnis der Wohlgeformtheitsprüfung, die durch den Wert von ENUMWellformedCheckType definiert ist, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
- `nType` ist der Wert von ENUMWellformedCheckType. Der Validierungstyp definiert, ob ein XML-Dokument oder eine DTD überprüft wird. Der Standardwert ist `eWellformedAny`.
- Bei einem Fehler wird eine `RaptorXMLException` ausgegeben. Mit Hilfe der LastErrorMessage Methode können Sie zusätzliche Informationen aufrufen.

Eigenschaften

Die Eigenschaften der **IXMLValidator** Schnittstelle sind im Folgenden in alphabetischer Reihenfolge beschrieben. Aus Gründen der Übersichtlichkeit sind die Eigenschaften in der Tabelle in Gruppen gegliedert. Beachten Sie, dass String-Inputs, die als URLs interpretiert werden sollen, als absolute Pfade angegeben werden müssen. Bei Verwendung eines relativen Pfads sollte im Aufrufmodul ein Mechanismus zum Auflösen des relativen Pfads definiert sein.

Datendateien	Schema-Dateien	Verarbeitung
InputFileArray	DTDFileName	AssessmentMode
InputTextArray	DTDFromText	EnableNamespaces
InputXMLFileName	SchemaFileArray	LastErrorMessage
InputXMLFromText	SchemaFileName	PythonScriptFile
	SchemaFromText	Streaming
	SchemaImports	XincludeSupport
	SchemalocationHints	XMLValidationMode
	SchemaMapping	XSDVersion
	SchemaTextArray	

ENUMAssessmentMode **AssessmentMode** [Nach oben | Methoden | Eigenschaften]
Definiert den Validierungsmodus des XML-Validierungsprozessors (strict oder lax), wie er durch die ENUMAssessmentMode Literale definiert ist.

---

string **DTDFileName** [Nach oben | Methoden | Eigenschaften]
Definiert, welches externe DTD-Dokument für die Validierung verwendet werden soll. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden DTD

angibt.

---

string **DTDFromText** [Nach oben | Methoden | Eigenschaften]
Liefert die gesamte DTD als String.

---

bool **EnableNamespaces** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Namespace-fähige Verarbeitung. Dies eignet sich zum Überprüfen der XML-Instanz auf Fehler infolge falscher Namespaces. Der Wert `true` aktiviert die Namespace-fähige Verarbeitung; `false` deaktiviert sie. Der Standardwert ist `false`.

---

object **InputFileArray** [Nach oben | Methoden | Eigenschaften]
Liefert ein Array der URLs der XML-Dateien, die als Input-Daten verwendet werden sollen. Die Eigenschaft liefert ein Objekt, das die absoluten URLs der einzelnen XML-Dateien in Form von Strings enthält.

---

object **InputTextArray** [Nach oben | Methoden | Eigenschaften]
Liefert ein Array der URLs der Textdateien, die als Input-Daten verwendet werden sollen. Die Eigenschaft liefert ein Objekt, das die absoluten URLs der einzelnen Textdateien in Form von Strings enthält.

---

string **InputXMLFileName** [Nach oben | Methoden | Eigenschaften]
Definiert die zu validierende XML-Datei. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden XML-Datei angibt.

---

string **InputXMLFromText** [Nach oben | Methoden | Eigenschaften]
Liefert einen Text-String mit dem Inhalt des zu validierenden XML-Dokuments.

---

string **LastErrorMessage** [Nach oben | Methoden | Eigenschaften]
Ruft die letzte Fehlermeldung des RaptorXML-Prozessors als String auf.

---

bool **ParallelAssessment** [Nach oben | Methoden | Eigenschaften]
Aktiviert/deaktiviert die parallele Validierung von Schemas.

---

string **PythonScriptFile** [Nach oben | Methoden | Eigenschaften]
Definiert die Python Script-Datei, die eine zusätzliche Verarbeitung der für die XML- oder XSD-Validierung bereitgestellten Datei liefert. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad des zu verwendenden Python Script angibt.

© 2014 Altova GmbH

object **SchemaFileArray** [Nach oben | Methoden | Eigenschaften]
Liefert ein Array der URLs der XSD-Dateien, die als externe XML-Schemas zu verwenden sind. Die Eigenschaft liefert ein Objekt, das die absoluten URLs der einzelnen XML-Schema-Dateien als Strings enthält.

string **SchemaFileName** [Nach oben | Methoden | Eigenschaften]
Definiert die externe XML-Schema-Datei, die für die Validierung verwendet werden soll. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden XML-Schema-Datei angibt.

string **SchemaFromText** [Nach oben | Methoden | Eigenschaften]
Liefert den Inhalt des XML-Schema-Dokuments, das für die Validierung verwendet werden soll, in Form eines Text-Strings.

ENUMSchemaImports **SchemaImports** [Nach oben | Methoden | Eigenschaften]
Definiert, wie Schemaimporte anhand der Attributwerte der `xs:import` Elemente behandelt werden sollen. Die Behandlung wird durch das ausgewählte ENUMSchemaImports Literal definiert.

ENUMLoadSchemalocation **SchemalocationHints** [Nach oben | Methoden | Eigenschaften]
Definiert, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll. Der Mechanismus wird durch das ausgewählte ENUMLoadSchemalocation Literal definiert.

ENUMSchemaMapping **SchemaMapping** [Nach oben | Methoden | Eigenschaften]
Definiert welches Mapping zum Auffinden des Schemas verwendet werden soll. Das Mapping wird durch das ausgewählte ENUMSchemaMapping Literal definiert.

object **SchemaTextArray** [Nach oben | Methoden | Eigenschaften]
Liefert ein Array von Strings, bei denen es sich um die XSD-Dateien handelt, die als externe XML-Schemas verwendet werden. Die Eigenschaft liefert ein Objekt, das die Textstrings der einzelnen XML-Schema-Dateien als Strings enthält.

bool **Streaming** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Streaming-Validierung. Im Streaming-Modus werden möglichst wenige Daten im Arbeitsspeicher gehalten, wodurch die Verarbeitung beschleunigt wird. Der Wert `true` aktiviert die Streaming-Validierung; `false` deaktiviert sie. Der Standardwert ist `true`.

bool **XincludeSupport** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Verwendung von XInclude Elementen. Der Wert true aktiviert die XInclude-Unterstützung; false deaktiviert sie. Der Standardwert ist false.

ENUMXMLValidationMode **XMLValidationMode** [Nach oben | Methoden | Eigenschaften]
Definiert den XML-Validierungsmodus (Validierung oder Wohlgeformtheitsprüfung). Der Modus ist derjenige, der durch das ENUMXMLValidationMode Literal definiert ist.

ENUMXSDVersion **XSDVersion** [Nach oben | Methoden | Eigenschaften]
Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert wird. Die Werte sind die ENUMXSDVersion Literale.

IXSLT

Die **IXSLT** Schnittstelle liefert Methoden und Eigenschaften zum Ausführen einer XSLT 1.0-, XSLT 2.0 oder XSLT 3.0-Transformation. Die Ausgabe kann als Datei gespeichert oder als Textstring zurückgegeben werden. Es können XSLT-Parameter an das XSLT-Stylesheet übergeben werden. Die URLs von XML- und XSLT-Dateien können über die Eigenschaften der Schnittstelle als Strings bereitgestellt werden. Alternativ dazu können das XML- und das XSLT-Dokument im Code als Textstrings konstruiert werden.

Anmerkung: Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Wenn ein relativer Pfad verwendet wird, sollte im aufrufenden Modul ein Mechanismus zur Auflösung des relativen Pfads definiert werden.

Anmerkung: Der XSLT 2.0- oder 3.0-Prozessor von RaptorXML kann zur Verarbeitung eines XSLT 1.0 Stylesheet im Rückwärtskompatibilitätsmodus verwendet werden. Die Ausgabe könnte sich allerdings von der Ausgabe, die anhand desselben XSLT 1.0 Stylesheet mit dem XSLT 1.0-Prozessor erzeugt wurde, unterscheiden.

Methoden
IsValid
Execute
ExecuteAndGetResultAsString
ExecuteAndGetResultAsStringWithBaseOutputURI
AddExternalParameter
ClearExternalParameterList

Eigenschaften		
ChartExtensionsEnabled	JavaBarcodeExtensionLocation	SchemaMapping
DotNetExtensionsEnabled	JavaExtensionsEnabled	StreamingSerialization

© 2014 Altova GmbH

EngineVersion	LastErrorMessage	XincludeSupport
IndentCharacters	LoadXMLWithPSVI	XMLValidationMode
InitialTemplateMode	NamedTemplateEntryPoint	XSDVersion
InputXMLFileName	SchemaImports	XSLFileName
InputXMLFromText	SchemalocationHints	XSLFromText

Methoden

Im Folgenden sind die Methoden der **IXSLT** Schnittstelle beschrieben. Beachten Sie, dass Strings, die als URLs interpretiert werden sollen, absolute Pfade liefern müssen. Wenn ein relativer Pfad verwendet wird, sollte im aufrufenden Modul ein Mechanismus zur Auflösung des relativen Pfads definiert werden.

Methoden
IsValid
Execute
ExecuteAndGetResultAsString
ExecuteAndGetResultAsStringWithBaseOutputURI
AddExternalParameter
ClearExternalParameterList

`bool` **IsValid()** [Nach oben | Methoden | Eigenschaften]

- Gibt das Ergebnis der Validierung des XSLT-Stylesheet, die anhand der in ENUMXSLTVersion (siehe ENUMXSLTVersion Eigenschaft) genannten XSLT-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der LastErrorMessage Operation können Sie zusätzliche Informationen aufrufen.

`bool` **Execute(string bstrResultFileName)** [Nach oben | Methoden | Eigenschaften]

- Führt die XSLT-Transformation anhand der in ENUMXSLTVersion (siehe ENUMXSLTVersion Eigenschaft) genannten XSLT-Spezifikation aus und speichert das Ergebnis in einer Ausgabedatei.
- Die Ausgabedatei wird durch `bstrResultFileName`, einen String, der die URL der Ausgabedatei bereitstellt, definiert.
- Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der LastErrorMessage Operation können Sie zusätzliche Informationen aufrufen.

`string` **`ExecuteAndGetResultAsString()`** [Nach oben | Methoden | Eigenschaften]

- Führt die XSLT-Transformation anhand der in `ENUMXSLTVersion` (siehe `EngineVersion` Eigenschaft) genannten XSLT-Spezifikation aus und gibt das Ergebnis als String zurück.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`string` **`ExecuteAndGetResultAsStringWithBaseOutputURI(`**`string bstrBaseURI`**`)`** [Nach oben | Methoden | Eigenschaften]

- Führt die XSLT-Transformation anhand der in `ENUMXSLTVersion` (siehe `EngineVersion` Eigenschaft) genannten XSLT-Spezifikation aus und gibt das Ergebnis unter dem durch die Basis-URI definierten Pfad (String `bstrBaseURI`) als String zurück.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`void` **`AddExternalParameter(`**`string bstrName, string bstrValue`**`)`** [Nach oben | Methoden | Eigenschaften]

- Fügt den Namen und Wert eines neuen externen Parameters hinzu. `bstrName` und `bstrValue` sind Strings.
- Jeder externe Parameter und sein Wert müssen in einem separaten Aufruf der Methode definiert werden. Die Parameter müssen im XSLT-Dokument, optional mit einer Typdeklaration deklariert sein. Unabhängig von der Typdeklaration im XSLT-Dokument wird kein spezielles Trennzeichen benötigt, wenn der Parameterwert mit `AddExternalParameter` bereitgestellt wird.

`void` **`ClearExternalParameterList()`** [Nach oben | Methoden | Eigenschaften]

- Löscht die Liste der mit der `AddExternalParameter` Methode erstellten externen Parameter.

Eigenschaften

Die Eigenschaften der **`IXSLT`** Schnittstelle sind im Folgenden in alphabetischer Reihenfolge beschrieben. Aus Gründen der Übersichtlichkeit sind die Eigenschaften in der Tabelle in Gruppen gegliedert. Beachten Sie, dass String-Inputs, die als URLs interpretiert werden sollen, als absolute Pfade angegeben werden müssen. Bei Verwendung eines relativen Pfads sollte im Aufrufmodul ein Mechanismus zum Auflösen des relativen Pfads definiert sein.

 © 2014 Altova GmbH

XML	XSLT	Schema
InputXMLFileName	EngineVersion	SchemaImports
InputXMLFromText	XSLFileName	SchemalocationHints
LoadXMLWithPSVI	XSLFromText	SchemaMapping
XincludeSupport		XSDVersion
XMLValidationMode		

Verarbeitung	Erweiterungen
IndentCharacters	ChartExtensionsEnabled
InitialTemplateMode	DotNetExtensionsEnabled
LastErrorMessage	JavaBarcodeExtensionLocation
NamedTemplateEntryPoint	JavaExtensionsEnabled
StreamingSerialization	

`bool` **ChartExtensionsEnabled** [Nach oben | Methoden | Eigenschaften]
Aktiviert bzw. deaktiviert Altova-Diagrammerweiterungsfunktionen. Der Wert `true` aktiviert Diagrammerweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

`bool` **DotNetExtensionsEnabled** [Nach oben | Methoden | Eigenschaften]
Aktiviert oder deaktiviert Visual Studio .NET-Erweiterungen. Der Wert `true` aktiviert .NET-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

ENUMXSLTVersion **EngineVersion** [Nach oben | Methoden | Eigenschaften]
Definiert die zu verwendende XSLT-Version (XSLT1.0/2.0/3.0). Der Eigenschaftswert ist ein ENUMXSLTVersion Literal.

`string` **IndentCharacters** [Nach oben | Methoden | Eigenschaften]
Definiert den als Einrückung zu verwendenden Zeichenstring.

`string` **InitialTemplateMode** [Nach oben | Methoden | Eigenschaften]
Definiert den Anfangsmodus für die XSLT-Verarbeitung. Vorlagen mit einem Moduswert, der dem bereitgestellten String entspricht, werden verarbeitet.

`string` **InputXMLFileName** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad der zu transformierenden XML-Input-Datei. Der bereitgestellte String muss eine

absolute URL sein, die den genauen Pfad zur zu verwendenden XML-Datei angibt.

`string` **InputXMLFromText** [Nach oben | Methoden | Eigenschaften]
Stellt den Inhalt des zu transformierenden XML-Dokuments als Textstring bereit.

`string` **JavaBarcodeExtensionLocation** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad der Java-Barcode-Erweiterungsdatei. Nähere Informationen dazu finden Sie im Abschnitt Barcode-Erweiterungsfunktionen von Altova. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad der zu verwendenden Datei angibt.

`bool` **JavaExtensionsEnabled** [Nach oben | Methoden | Eigenschaften]
Aktiviert oder deaktiviert Java-Erweiterungen. Der Wert `true` aktiviert Java-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

`string` **LastErrorMessage** [Nach oben | Methoden | Eigenschaften]
Ruft die letzte Fehlermeldung vom RaptorXML-Prozessor als String auf.

`bool` **LoadXMLWithPSVI** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Option zum Laden und Verwenden des Post Schema Validation Infoset (PSVI). Wenn das PSVI geladen ist, können Daten im XML-Dokument anhand von Informationen aus dem Schema qualifiziert werden. Der Wert `true` aktiviert das Laden des PSVI; `false` deaktiviert das Laden des PSVI.

`string` **NamedTemplateEntryPoint** [Nach oben | Methoden | Eigenschaften]
Definiert den Namen der benannten Vorlage, die als Einstiegspunkt für die Transformation verwendet werden soll, als String.

ENUMSchemaImports **SchemaImports** [Nach oben | Methoden | Eigenschaften]
Definiert, wie Schemaimporte anhand der Attributwerte der `xs:import` Elemente behandelt werden sollen. Die Behandlung wird durch das ausgewählte ENUMSchemaImports Literal definiert.

ENUMLoadSchemalocation **SchemalocationHints** [Nach oben | Methoden | Eigenschaften]
Definiert, welcher Mechanismus zum Auffinden des Schemas verwendet werden soll. Der Mechanismus wird durch das ausgewählte ENUMLoadSchemalocation Literal definiert.

ENUMSchemaMapping **SchemaMapping** [Nach oben | Methoden | Eigenschaften]

 © 2014 Altova GmbH

Definiert, welches Mapping zum Auffinden des Schemas verwendet werden soll. Das Mapping wird durch das ausgewählte ENUMSchemaMapping Literal definiert.

---

bool **StreamingSerialization** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Streaming-Serialisierung. Im Streaming-Modus werden möglichst wenige Daten im Arbeitsspeicher gehalten, wodurch die Verarbeitung beschleunigt wird. Der Wert `true` aktiviert die Streaming-Serialisierung; `false` deaktiviert sie.

---

bool **XincludeSupport** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Verwendung von `XInclude` Elementen. Der Wert `true` aktiviert die XInclude-Unterstützung; `false` deaktiviert sie. Der Standardwert ist `false`.

ENUMXMLValidationMode **XMLValidationMode** [Nach oben | Methoden | Eigenschaften]
Definiert den XML-Validierungsmodus (Validierung oder Wohlgeformtheitsprüfung). Der Modus wird durch das ausgewählte ENUMXMLValidationMode Literal definiert.

ENUMXSDVersion **XSDVersion** [Nach oben | Methoden | Eigenschaften]
Definiert die XML-Schema-Version, anhand welcher das XML-Dokument validiert werden soll. Die Werte sind die ENUMXSDVersion Literale.

---

string **XSLFileName** [Nach oben | Methoden | Eigenschaften]
Definiert die für die Transformation zu verwendende XSLT-Datei. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur zu verwendenden XSLT-Datei angibt.

---

string **XSLFromText** [Nach oben | Methoden | Eigenschaften]
Liefert den Inhalt des für die Transformation zu verwendenden XSLT-Dokuments als Textstring.

IXQuery

Die **IXQuery**-Schnittstelle stellt Methoden und Eigenschaften zur Ausführung eines XQuery 1.0- oder 3.0-Dokuments zur Verfügung. Die Ergebnisse können in einer Datei gespeichert oder als String zurückgegeben werden. Mit dem Objekt können auch externe XQuery-Variablen an das XQuery-Dokument übergeben werden. Die URLs von XQuery- und XML-Dateien können über die Eigenschaften der Schnittstelle als Strings bereitgestellt werden. Alternativ dazu können XML- und XQuery-Dokumente im Code in Form von Textstrings erstellt werden.

Anmerkung: Wo String-Inputs als URLs interpretiert werden sollen, sollten absolute Pfade verwendet werden. Wenn ein relativer Pfad verwendet wird, sollte im aufrufenden Modul ein Mechanismus zur Auflösung des relativen Pfads definiert werden.

Methoden

IsValid
IsValidUpdate
Execute
ExecuteUpdate
ExecuteAndGetResultAsString
ExecuteUpdateAndGetResultAsString
AddExternalVariable
ClearExternalParameterList

Eigenschaften			
ChartExtensionsEnabled	InputXMLFromText	OutputEncoding	XMLValidationMode
DotNetExtensionsEnabled	JavaBarcodeExtensionLocation	OutputIndent	XQueryFileName
EngineVersion	JavaExtensionsEnabled	OutputMethod	XQueryFromText
IndentCharacters	LastErrorMessage	OutputOmitXMLDeclaration	XSDVersion
InputXMLFileName	LoadXMLWithPSVI	XincludeSupport	

Methoden

Im Folgenden sind die Methoden der **`IXQuery`** Schnittstelle beschrieben. Beachten Sie, dass Strings, die als URLs interpretiert werden sollen, absolute Pfade liefern müssen. Wenn ein relativer Pfad verwendet wird, sollte im aufrufenden Modul ein Mechanismus zur Auflösung des relativen Pfads definiert werden.

Methoden
IsValid
IsValidUpdate
Execute
ExecuteUpdate
ExecuteAndGetResultAsString
ExecuteUpdateAndGetResultAsString
AddExternalVariable
ClearExternalParameterList

 © 2014 Altova GmbH

`bool` **`IsValid()`** [Nach oben | Methoden | Eigenschaften]

- Gibt das Ergebnis der Validierung des XQuery-Stylesheet, die anhand der in `ENUMXQueryVersion` (siehe `EngineVersion` Eigenschaft) genannten XQuery-Spezifikation durchgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`bool` **`IsValidUpdate()`** [Nach oben | Methoden | Eigenschaften]

- Gibt das Ergebnis der Validierung des XQuery Update-Dokuments, das entsprechend der in `ENUMXQueryVersion` (siehe `EngineVersion` Eigenschaft) genannten XQuery Update-Spezifikation ausgeführt wurde, zurück. Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Methode können Sie zusätzliche Informationen aufrufen.

`bool` **`Execute(string bstrOutputFile)`** [Nach oben | Methoden | Eigenschaften]

- Führt das XQuery-Dokument anhand der in `ENUMXQueryVersion` (siehe `EngineVersion` Eigenschaft) genannten XQuery-Spezifikation aus und speichert das Ergebnis in einer Ausgabedatei.
- Die Ausgabedatei wird durch `bstrOutputFile`, einen String, der die URL der Ausgabedatei bereitstellt, definiert.
- Das Ergebnis ist bei Erfolg `true`, bei Fehlschlag `false`.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`bool` **`ExecuteUpdate(string bstrOutputFile)`** [Nach oben | Methoden | Eigenschaften]

- Führt das XQuery Update entsprechend der in `ENUMXQueryVersion` (siehe `EngineVersion` Eigenschaft) genannten XQuery Update-Spezifikation aus und speichert das Ergebnis in einer Ausgabedatei.
- Die Ausgabedatei wird durch `bstrOutputFile` definiert, welches ein String ist, der die URL, der Ausgabedatei angibt.
- Bei erfolgreicher Ausführung wird der Boolesche Wert `true` zurückgegeben, bei Fehlschlag der Wert `false`.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`string` **`ExecuteAndGetResultAsString()`** [Nach oben | Methoden | Eigenschaften]

- Führt das XQuery-Dokument anhand der in `ENUMXQueryVersion` (siehe `EngineVersion` Eigenschaft) genannten XQuery-Spezifikation aus und gibt das Ergebnis als String zurück.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`string` **`ExecuteUpdateAndGetResultAsString()`** Nach oben | Methoden | Eigenschaften]

- Führt das XQuery Update-Dokument anhand der in `ENUMXQueryVersion` (siehe `EngineVersion` Eigenschaft) genannten XQuery Update-Spezifikation aus und gibt das Transformationsergebnis als String zurück.
- Bei Auftreten eines Fehlers wird eine `RaptorXMLException` ausgelöst. Mit Hilfe der `LastErrorMessage` Operation können Sie zusätzliche Informationen aufrufen.

`void` **`AddExternalVariable(`**`string bstrName, string bstrValue`**`)`** [Nach oben | Methoden | Eigenschaften]

- Fügt den Namen und Wert einer externen Variablen hinzu. `bstrName` und `bstrValue` sind Strings.
- Jede externe Variable und ihr Wert müssen in einem separaten Aufruf der Methode definiert werden. Die Variablen müssen im XQuery-Dokument, optional mit einer Typdeklaration deklariert sein. Setzen Sie den Wert der Variablen in einfache Anführungszeichen, wenn es sich dabei um einen String handelt.

`void` **`ClearExternalVariableList()`** [Nach oben | Methoden | Eigenschaften]

- Löscht die Liste der mit der `AddExternalVariable` Methode erstellten externen Variablen.

Eigenschaften

Die Eigenschaften der **`IXQuery`** Schnittstelle sind im Folgenden in alphabetischer Reihenfolge beschrieben. Aus Gründen der Übersichtlichkeit sind die Eigenschaften in der Tabelle in Gruppen gegliedert. Beachten Sie, dass String-Inputs, die als URLs interpretiert werden sollen, als absolute Pfade angegeben werden müssen. Bei Verwendung eines relativen Pfads sollte im Aufrufmodul ein Mechanismus zum Auflösen des relativen Pfads definiert sein.

 © 2014 Altova GmbH

XML	XQuery	Verarbeitung	Erweiterungen
InputXMLFileName	EngineVersion	IndentCharacters	ChartExtensionsEnabled
KeepFormatting	XQueryFileName	LastErrorMessage	DotNetExtensionsEnabled
InputXMLFromText	XQueryFromText	OutputEncoding	JavaBarcodeExtensionLocation
LoadXMLWithPSVI		OutputIndent	JavaExtensionsEnabled
XincludeSupport		OutputMethod	
XMLValidationMode		OutputOmitXMLDeclaration	
XSDVersion		UpdatedXMLWriteMode	

`bool` **ChartExtensionsEnabled** [Nach oben | Methoden | Eigenschaften]
Aktiviert bzw. deaktiviert Altova-Diagrammerweiterungsfunktionen. Der Wert `true` aktiviert Diagrammerweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

`bool` **DotNetExtensionsEnabled** [Nach oben | Methoden | Eigenschaften]
Aktiviert oder deaktiviert Visual Studio .NET-Erweiterungen. Der Wert `true` aktiviert .NET-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

ENUMXQueryVersion **EngineVersion** [Nach oben | Methoden | Eigenschaften]
Definiert die zu verwendende XQuery-Version (1.0 oder 3.0). Der Eigenschaftswert ist ein ENUMXQueryVersion Literal.

`string` **IndentCharacters** [Nach oben | Methoden | Eigenschaften]
Definiert den als Einrückung zu verwendenden Zeichenstring.

`string` **InputXMLFileName** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad der zu verarbeitenden XML-Datei. Der bereitgestellte String muss eine absolute URL sein, die den genauen Pfad zur zu verwendenden XML-Datei angibt.

`string` **InputXMLFromText** [Nach oben | Methoden | Eigenschaften]
Stellt den Inhalt des zu verarbeitenden XML-Dokuments als Textstring bereit.

`string` **JavaBarcodeExtensionLocation** [Nach oben | Methoden | Eigenschaften]
Definiert den Pfad der Java-Barcode-Erweiterungsdatei. Nähere Informationen dazu finden Sie im Abschnitt Barcode-Erweiterungsfunktionen von Altova. Der bereitgestellte String muss eine

absolute URL sein, die den Basispfad der zu verwendenden Datei angibt.

`bool` **JavaExtensionsEnabled** [Nach oben | Methoden | Eigenschaften]
Aktiviert oder deaktiviert Java-Erweiterungen. Der Wert `true` aktiviert Java-Erweiterungen; `false` deaktiviert sie. Der Standardwert ist `true`.

`bool` **KeepFormatting** [Nach oben | Methoden | Eigenschaften]
Definiert, ob die Formatierung des Originaldokuments so weit wie möglich beibehalten werden soll oder nicht. Beim Wert `true` wird die Formatierung beibehalten, bei `false` wird sie nicht beibehalten. Der Standardwert ist `true`.

`string` **LastErrorMessage** [Nach oben | Methoden | Eigenschaften]
Ruft die letzte Fehlermeldung des RaptorXML-Prozessors als String auf.

`bool` **LoadXMLWithPSVI** [Nach oben | Methoden | Eigenschaften]
Aktiviert die Option zum Laden und Verwenden des Post Schema Validation Infoset (PSVI). Wenn das PSVI geladen ist, können Daten im XML-Dokument anhand von Informationen aus dem Schema qualifiziert werden. Der Wert `true` aktiviert das Laden des PSVI; `false` deaktiviert das Laden des PSVI.

`string` **OutputEncoding** [Nach oben | Methoden | Eigenschaften]
Definiert die Kodierung für das Ergebnisdokument. Verwenden Sie den Namen einer offiziellen IANA-Kodierung wie z.B. `UTF-8`, `UTF-16`, `US-ASCII`, `ISO-8859-1` als String.

`bool` **OutputIndent** [Nach oben | Methoden | Eigenschaften]
Aktiviert bzw. deaktiviert die Einrückung des Ausgabedokuments. Der Wert `true` aktiviert die Einrückung; `false` deaktiviert sie.

`string` **OutputMethod** [Nach oben | Methoden | Eigenschaften]
Definiert die Serialisierung des Ausgabedokuments. Gültige Werte sind: `xml | xhtml | html | text`. Der Standardwert ist `xml`.

`bool` **OutputOmitXMLDeclaration** [Nach oben | Methoden | Eigenschaften]
Aktiviert/deaktiviert den Einschluss der XML-Deklaration in das Ergebnisdokument. Der Wert `true` lässt die Deklaration weg; `false` inkludiert sie. Der Standardwert ist `false`.

 © 2014 Altova GmbH

ENUMXQueryUpdatedXML **UpdatedXMLWriteMode** [Nach oben | Methoden | Eigenschaften]
Definiert, wie Aktualisierungen der XML-Datei gehandhabt werden. Die Eigenschaftswert ist ein ENUMXQueryUpdatedXML Literal.

---

bool **XincludeSupport** [Nach oben | Methoden | Eigenschaften]
Aktiviert bzw. deaktiviert die Verwendung von XInclude Elementen. Der Wert true aktiviert die XInclude-Unterstützung; false deaktiviert sie. Der Standardwert ist false.

ENUMXMLValidationMode **XMLValidationMode** [Nach oben | Methoden | Eigenschaften]
Definiert den XML-Validierungsmodus (Validierung oder Wohlgeformtheitsprüfung). Der Modus ist derjenige, der durch das ENUMXMLValidationMode Literal definiert ist.

---

string **XQueryFileName** [Nach oben | Methoden | Eigenschaften]
Definiert die zu verwendende XQuery-Datei. Der bereitgestellte String muss eine absolute URL sein, die den Basispfad des zu verwendenden XSLT-Datei angibt.

---

string **XQueryFromText** [Nach oben | Methoden | Eigenschaften]
Liefert den Inhalt des zu verwendenden XQuery-Dokuments in Form eines Text-Strings.

ENUMXSDVersion **XSDVersion** [Nach oben | Methoden | Eigenschaften]
Definiert das XML-Schema-Dokument, das für die Validierung des XML-Dokuments verwendet werden soll. Die Werte sind die ENUMXSDVersion Literale.

7.4.2 Enumerationen

Es sind die folgenden Enumerationen definiert. Eine Beschreibung dazu finden Sie in den Unterabschnitten dieses Abschnitts.

ENUMAssessmentMode

ENUMErrorFormat

ENUMLoadSchemalocation

ENUMQueryVersion

ENUMSchemaImports

ENUMSchemaMapping

ENUMValidationType

ENUMWellformedCheckType

ENUMXMLValidationMode

ENUMXQueryVersion

ENUMXSDVersion

ENUMXSLTVersion

ENUMAssessmentMode

Beschreibung

Enthält Enumerationsliterale, die den XML-Validierungsmodus definieren: `Strict` oder `Lax`.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	AssessmentMode

Enumerationsliterale

```
eAssessmentModeStrict   = 0
eAssessmentModeLax      = 1
```

`eAssessmentModeStrict`

Setzt den Schema-Validierungsmodus auf `Strict`. Dies ist der Standardwert.

`eAssessmentModeLax`

Setzt den Schema-Validierungsmodus auf `Lax`.

 © 2014 Altova GmbH

ENUMErrorFormat

Beschreibung
Enthält Enumerationsliterale, die das Format der Fehlerausgabe definieren.

Verwendet von

Schnittstelle	*Operation*
IServer	ErrorFormat

Enumerationsliterale

```
eFormatText          = 0
eFormatShortXML      = 1
eFormatLongXML       = 2
```

eFormatText
Definiert das Fehlerausgabeformat als `Text`. Dies ist der Standardwert.

eFormatShortXML
Definiert das Fehlerausgabeformat als `ShortXML`. Dieses Format ist eine Kurzform des Formats `LongXML`.

eFormatLongXML
Definiert das Fehlerausgabeformat als `LongXML`. Dieses Format ist das ausführlichste Fehlerausgabeformat.

ENUMLoadSchemalocation

Beschreibung
Enthält Enumerationsliterale, die angeben, wie der Schemapfad ermittelt werden soll.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	SchemalocationHints
IXSLT	SchemalocationHints

Enumerationsliterale

```
eSHLoadBySchemalocation     = 0
eSHLoadByNamespace          = 1
eSHLoadCombiningBoth        = 2
eSHLoadIgnore               = 3
```

eSHLoadBySchemalocation
Setzt das Laden des Schemas auf `LoadBySchemalocation`. Verwendet die URL des Schemapfads in den Attributen `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` in XML- oder XBRL-Instanzattributen. Dies ist der **Standardwert**.

eSHLoadByNamespace
Setzt das Laden des Schemas auf `LoadByNamespace`. Verwendet den Namespace-Teil von `xsi:schemaLocation` und im Fall von `xsi:noNamespaceSchemaLocation` einen leeren String und sucht das Schema über ein Katalogmapping.

eSHLoadCombiningBoth
Setzt das Laden des Schemas auf `CombiningBoth`. Wenn entweder der Namespace oder die URL ein Katalogmapping hat, so wird das Katalogmapping verwendet. Haben beide Katalogmappings, hängt es vom Wert des `ENUMSchemaMapping`-Parameters ab, welches Mapping verwendet wird. Wenn weder der Namespace noch die URL ein Katalogmapping hat, wird die URL verwendet.

eSHLoadIgnore
Setzt das Laden des Schemas auf `LoadIgnore`. Wenn der Wert des Parameters `eSHLoadIgnore` ist, werden die Attribute `xsi:schemaLocation` und `xsi:noNamespaceSchemaLocation` beide ignoriert.

ENUMQueryVersion

Beschreibung
Enthält die Enumerationsliterale, die die zu verwendende XQuery-Version (XQuery 1.0 oder 3.0) definieren.

Enumerationsliterale

```
eXQVersion10        = 1
eXQVersion30        = 3
```

eXQVersion10
Definiert XQuery 1.0 als die zu verwendende XQuery-Version.

 © 2014 Altova GmbH

eXQVersion30
Definiert XQuery 3.0 als die zu verwendende XQuery-Version.

ENUMSchemaImports

Beschreibung
Enthält die Enumerationsliterale, die das Verhalten von `xs:import`-Elementen, von denen jedes ein optionales `namespace`-Attribut und ein optionales `schemaLocation`-Attribut hat, definieren.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	SchemaImports
IXSLT	SchemaImports

Enumerationsliterale

```
eSILoadBySchemalocation           = 0
eSILoadPreferringSchemalocation   = 1
eSILoadByNamespace                = 2
eSICombiningBoth                  = 3
eSILicenseNamespaceOnly           = 4
```

eSILoadBySchemalocation
Setzt den Schema-Import auf `LoadBySchemalocation`. Das Schema wird unter Berücksichtigung von Katalogmappings anhand des Werts des `schemaLocation`-Attributs gesucht. Ist das `namespace`-Attribut vorhanden, wird der Namespace importiert (lizenziert).

eSILoadPreferringSchemalocation
Setzt den Schema-Import auf `LoadPreferringSchemalocation`. Ist das Attribut `schemaLocation` vorhanden, wird es verwendet, wobei Katalogmappings berücksichtigt werden. Ist kein `schemaLocation`-Attribut vorhanden, so wird der Wert des `namespace`-Attributs über ein Katalogmapping verwendet. Dieses Literal ist der **Standardwert** der Enumeration.

eSILoadByNamespace
Setzt den Schema-Import auf `LoadByNamespace`. Das Schema wird anhand des Werts des `namespace`-Attributs über ein Katalogmapping gesucht.

eSICombiningBoth
Setzt den Schema-Import auf `CombiningBoth`. Wenn entweder das `namespace`-Attribut oder das `schemaLocation`-Attribut ein Katalogmapping hat, so wird das Katalogmapping verwendet. Wenn

beide Attribute Katalogmappings haben, hängt es vom ENUMSchemaMapping-Parameter ab, welches Mapping verwendet wird. Wenn kein Katalogmapping vorhanden ist, wird das schemaLocation-Attribut (welches eine URL sein sollte) verwendet.

eSILicenseNamespaceOnly
Setzt den Schema-Import auf LicenseNamespaceOnly. Der Namespace wird importiert. Es wird kein Schema-Dokument importiert.

ENUMSchemaMapping

Beschreibung
Enthält die Enumerationsliterale, die definieren, welches der beiden Katalogmappings verwendet wird: Namespace- oder Schemapfad-URLs. Diese Enumeration eignet sich, um ENUMLoadSchemalocation und ENUMSchemaImports eindeutig zu unterscheiden.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	SchemaMapping
IXSLT	SchemaMapping

Enumerationsliterale

```
eSMPreferSchemalocation     = 0
eSMPreferNamespace          = 1
```

eSMPreferSchemalocation
Definiert, dass die Schema-Mapping-Option die Schemapfad-URI auswählt.

eSMPreferNamespace
Definiert, dass die Schema-Mapping-Option den Namespace auswählt.

ENUMValidationType

Beschreibung
Enthält Enumerationsliterale, die den Typ des zu validierenden Dokuments definieren.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	IsValid

 © 2014 Altova GmbH

Enumerationsliterale

```
eValidateAny        = 0
eValidateXMLWithDTD = 1
eValidateXMLWithXSD = 2
eValidateDTD        = 3
eValidateXSD        = 4
```

eValidateAny
Setzt den Validierungstyp auf `Any`. Validiert ein Dokument, nachdem sein Typ automatisch ermittelt wurde.

eValidateXMLWithDTD
Setzt den Validierungstyp auf `XMLWithDTD`. Dies definiert, dass ein XML-Dokument anhand einer DTD validiert wird.

eValidateXMLWithXSD
Setzt den Validierungstyp auf `XMLWithXSD`. Dies definiert, dass ein XML-Dokument anhand eines XML-Schemas validiert wird.

eValidateDTD
Setzt den Validierungstyp auf `ValidateDTD`. Dies definiert die Validierung eines DTD-Dokuments.

eValidateXSD
Setzt den Validierungstyp auf `ValidateXSD`. Dies definiert die Validierung eines W3C XML Schema-Dokuments.

ENUMWellformedCheckType

Beschreibung
Enthält die Enumerationsliterale, die den Typ des zu prüfenden Dokuments definieren: XML oder DTD.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	IsWellFormed

Enumerationsliterale

```
eWellFormedAny      = 0
eWellFormedXML      = 1
eWellFormedDTD      = 2
```

eWellformedAny
Setzt die Art der Wohlgeformtheitsprüfung auf `Any`. Damit wird ein XML- oder DTD-Dokument auf Wohlgeformtheit geprüft, nachdem automatisch ermittelt wurde, um welchen der beiden Dokumenttypen es sich handelt.

eWellformedXML
Setzt die Art der Wohlgeformtheitsprüfung auf `XML`. Damit wird ein XML-Dokument entsprechend der XML 1.0- oder XML 1.1-Spezifikation auf Wohlgeformtheit geprüft.

eWellformedDTD
Setzt die Art der Wohlgeformtheitsprüfung auf `DTD`. Damit wird ein DTD-Dokument auf Wohlgeformtheit geprüft.

ENUMXMLValidationMode

Beschreibung
Enthält die Enumerationsliterale, die festlegen, welche Art der XML-Verarbeitung verwendet wird: Validierung oder Wohlgeformtheit.

Verwendet von

Schnittstelle	*Operation*
IXMLValidator	XMLValidationMode
IXQuery	XMLValidationMode
IXSLT	XMLValidationMode

Enumerationsliterale

```
eXMLValidationModeWF     = 0
eXMLValidationModeID     = 1
eXMLValidationModeValid  = 2
```

eXMLValidationModeWF
Setzt den XML-Verarbeitungsmodus auf `Wellformed`. Dies ist der Standardwert.

eXMLValidationModeID
Intern.

 © 2014 Altova GmbH

eXMLValidationModeValid
Setzt den XML-Verarbeitungsmodus auf `Validation`.

ENUMXQueryVersion

Beschreibung
Enthält Enumerationsliterale, die die zu verwendende XQuery-Version definieren: XQuery 1.0 oder 3.0.

Verwendet von

Schnittstelle	*Operation*
IXQuery	EngineVersion

Enumerationsliterale

```
eXQVersion10        = 1
eXQVersion30        = 3
```

eXQVersion10
Definiert XQuery 1.0 als die zu verwendende XQuery-Version.

eXQVersion30
Definiert XQuery 3.0 als die zu verwendende XQuery-Version. Dies ist der Standardwert.

ENUMXQueryUpdatedXML

Beschreibung
Enthält Enumerationsliterale, die angeben, wie XQuery Updates gehandhabt werden.

Verwendet von

Schnittstelle	*Operation*
IXQuery	UpdatedXMLWriteMode

Enumerationsliterale

```
eUpdatedDiscard         = 1
eUpdatedWriteback       = 2
eUpdatedAsMainResult    = 3
```

eUpdatedDiscard
Aktualisierungen werden verworfen und nicht in eine Datei geschrieben.

eUpdatedWriteback
Aktualisierungen werden in die mit `InputXMLFileName` definierte XML-Input-Datei geschrieben.

eUpdatedAsMainResult
Aktualisierungen werden in die Datei geschrieben, die durch den Parameter `outputFile` von `ExecuteUpdate` definiert ist.

ENUMXSDVersion

Beschreibung
Enthält Enumerationsliterale, die die XML-Schema-Version definieren, anhand der das Dokument validiert werden soll: XSD 1.0 oder 1.1.

Verwendet von

Schnittstelle	*Operation*
`IXMLValidator`	`XSDVersion`
`IXQuery`	`XSDVersion`
`IXSLT`	`XSDVersion`

Enumerationsliterale

```
eXSDVersionAuto     = 0
eXSDVersion10       = 1
eXSDVersion11       = 2
```

eXSDVersionAuto
Setzt die für die Dokumentvalidierung zu verwendende XML-Schema-Version auf `Auto-detect`. Die XSD-Version wird automatisch nach Parsen des XSD-Dokuments ermittelt. Wenn das `vc:minVersion` Attribut des XSD-Dokuments den Wert `1.1` hat, wird das Dokument als XSD 1.1 erkannt. Wenn das Attribut einen anderen Wert hat oder fehlt, wird das Dokument als XSD 1.0 erkannt.

eXSDVersion10
Setzt die für die Dokumentvalidierung zu verwendende XML-Schema-Version auf XML-Schema 1.0.

 © 2014 Altova GmbH

eXSDVersion11
Setzt die für die Dokumentvalidierung zu verwendende XML-Schema-Version auf XML-Schema 1.1.

ENUMXSLTVersion

Beschreibung
Enthält Enumerationsliterale, die festlegen, welche XSLT-Version verwendet werden soll: XSLT 1.0, 2.0 oder 3.0.

Verwendet von

Schnittstelle	*Operation*
IXSLT	EngineVersion

Enumerationsliterale

```
eVersion10     = 1
eVersion20     = 2
eVersion30     = 3
```

eVersion10
Definiert XSLT 1.0 als die zu verwendende XSLT-Version.

eVersion20
Definiert XSLT 2.0 als die zu verwendende XSLT-Version.

eVersion30
Definiert XSLT 3.0 als die zu verwendende XSLT-Version.

Kapitel 8

Informationen zum XSLT- und XQuery-Prozessor

8 Informationen zum XSLT- und XQuery-Prozessor

Der XSLT- und der XQuery-Prozessor von RaptorXML Server hält sich genau an die W3C-Spezifikationen und ist daher strenger als die früheren Altova-Prozessoren, wie z.B. die in frühere Versionen von XMLSpy integrierten und die in AltovaXML, das Vorgängerprodukt von RaptorXML, integrierten. Infolgedessen werden auch leichte Fehler, die von früheren Prozessoren ignoriert wurden, von RaptorXML Server als Fehler gekennzeichnet.

Zum Beispiel:

- Wenn das Ergebnis eines Pfad-Operators sowohl Nodes als auch Nicht-Nodes enthält, wird ein Typfehler (`err:XPTY0018`) ausgegeben.
- Wenn `E1` in einem Pfadausdruck `E1/E2` nicht zu einer Node-Sequenz ausgewertet wird, wird ein Typfehler (`err:XPTY0019`) ausgegeben.

Ändern Sie bei Auftreten eines solchen Fehlers je nach Bedarf, entweder das XSLT/XQuery-Dokument oder das Instanzdokument.

In diesem Abschnitt sind implementierungsspezifische Funktionalitäten der Prozessoren geordnet nach Spezifikation beschrieben:

- XSLT 1.0
- XSLT 2.0
- XSLT 3.0
- XQuery 1.0
- XQuery 3.0

© 2014 Altova GmbH

8.1 XSLT 1.0

Der XSLT 1.0-Prozessor von RaptorXML Server entspricht der XSLT 1.0 Recommendation vom 16. November 1999 und der XPath 1.0 Recommendation vom 16. November 1999 des World Wide Web Consortium (W3C). Beachten Sie die folgenden Informationen zur Implementierung.

Anmerkungen zur Implementierung

Wenn das `method`-Attribut von `xsl:output` auf HTML gesetzt ist oder wenn standardmäßig die HTML-Ausgabe ausgewählt ist, werden Sonderzeichen in der XML- oder XSLT-Datei als HTML-Zeichenreferenzen in das HTML-Ausgabedokument eingefügt. So wird z.B. das Zeichen ` ` (die Dezimalzeichen-Referenz für ein geschütztes Leerzeichen) als ` ` in den HTML-Code eingefügt.

8.2 XSLT 2.0

In diesem Abschnitt:

- Prozessorkonformität
- Rückwärtskompatibilität
- Namespaces
- Schema-Fähigkeit
- Implementierungsspezifisches Verhalten

Konformität
Der XSLT 2.0-Prozessor von RaptorXML Server entspricht der XSLT 2.0 Recommendation vom 23. Jänner 2007 und der XPath 2.0 Recommendation vom 14. Dezember 2010 des World Wide Web Consortium (W3C).

Rückwärtskompatibilität
Der XSLT 2.0-Prozessor ist rückwärtskompatibel. Die Rückwärtskompatibilität des XSLT 2.0.-Prozessors kommt nur dann zum Einsatz, wenn Sie den XSLT 2.0-Prozessor (CLI-Parameter `--xslt=2`) zur Verarbeitung eines XSLT 1.0 Stylesheets verwenden. Beachten Sie, dass sich das Ergebnis des XSLT 1.0-Prozessors und des rückwärtskompatiblen XSLT 2.0.Prozessors unter Umständen unterscheiden kann.

Namespaces
In Ihrem XSLT 2.0 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie die in XSLT 2.0 verfügbaren Typ-Konstruktoren und Funktionen verwenden können. Normalerweise werden die unten aufgelisteten Präfixe verwendet; bei Bedarf können Sie auch andere Präfixe verwenden.

Namespace Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`

Normalerweise werden diese Namespaces im Element `xsl:stylesheet` oder `xsl:transform` deklariert, wie unten gezeigt:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
```

 © 2014 Altova GmbH

```
    ...
</xsl:stylesheet>
```

Beachten Sie bitte die folgenden Punkte:

- Der XSLT 2.0-Prozessor verwendet als Standard-Funktions-Namespace den Namespace für XPath 2.0- und XQuery 1.0-Funktionen (siehe Tabelle oben). Sie können daher XPath 2.0- und XSLT 2.0-Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.
- Bei Verwendung von Typ-Konstruktoren und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. xs:date) das in der jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- Einige XPath 2.0-Funktionen haben denselben Namen wie XML Schema-Datentypen. So gibt es z.B. für die XPath-Funktionen `fn:string` und `fn:boolean` XML-Schema-Datentypen mit demselben lokalen Namen: `xs:string` und `xs:boolean`. Wenn Sie daher den XPath-Ausdruck `string('Hello')` verwenden, wird der Ausdruck als `fn:string('Hello')` ausgewertet und nicht als `xs:string('Hello')`.

Schemafähigkeit
Der XSLT 2.0-Prozessor ist schemafähig. Sie können daher benutzerdefinierte Schematypen und die `xsl:validate`-Anweisung verwenden.

Implementierungsspezifisches Verhalten
Im Folgenden finden Sie eine Beschreibung, wie der XSLT 2.0-Prozessor implementierungsspezifische Aspekte des Verhaltens bestimmter XSLT 2.0-Funktionen behandelt.

`xsl:result-document`
Zusätzlich werden die folgenden Kodierungen unterstützt: `x-base16tobinary` und `x-base64tobinary`.

`function-available`
Die Funktion überprüft, ob in-scope-Funktionen (XSLT 2.0, XPath 2.0 und Erweiterungsfunktionen) verfügbar sind.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden Kodierungen unterstützt: `binarytobase16` und `binarytobase64`.

`unparsed-text-available`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die Kodierungen `x-binarytobase16` und `x-binarytobase64` unterstützt.

Anmerkung: Die folgenden Kodierungswerte, die in früheren Versionen von AltovaXML, dem nicht mehr vertriebenen Altova-Prozessor zur Validierung von XML-Dateien und Transformationen von XSLT/XQuery-Dateien verwendet wurden, werden nun nicht mehr verwendet:`base16tobinary`, `base64tobinary`, `binarytobase16` und `binarytobase64`.

8.3 XSLT 3.0

Der XSLT 3.0-Prozessor von RaptorXML Server entspricht dem XSLT 3.0 Last Call Working Draft vom 12. Dezember 2013 und der XPath 3.0 Recommendation vom 8. April 2014 des World Wide Web Consortium (W3C).

Der XSLT 3.0-Prozessor hat dieselben implementierungsspezifischen Eigenschaften wie der XSLT 2.0-Prozessor. Zusätzlich dazu unterstützt er die folgenden XSLT 3.0-Funktionen: `xsl:evaluate`, `xsl:try`, `xsl:catch`, `xsl:map`, `xsl:map-entry`, Textwertvorlagen, XPath- und XQuery 3.0-Funktionen und Operatoren und die XPath 3.0-Spezifikation.

Die folgenden XSLT 3.0-Anweisungen werden derzeit nicht unterstützt:

```
xsl:accept
xsl:accumulator
xsl:accumulator-rule
xsl:assert
xsl:break
xsl:context-item
xsl:expose
xsl:fork
xsl:iterate
xsl:merge
xsl:merge-action
xsl:merge-key
xsl:merge-source
xsl:mode
xsl:next-iteration
xsl:next-match
xsl:on-completion
xsl:override
xsl:package
xsl:stream
xsl:use-package
```

8.4 XQuery 1.0

In diesem Abschnitt:

- Standardkonformität
- Schema-Fähigkeit
- Kodierung
- Namespaces
- XML-Quelle und Validierung
- Statische und dynamische Typüberprüfung
- Bibliotheksmodule
- Externe Module
- Collations
- Präzision von numerischen Daten
- Unterstützung für XQuery-Anweisungen

Standardkonformität
Der XQuery 1.0-Prozessor von RaptorXML Server entspricht der XQuery 1.0 Recommendation vom 14. Dezember 2010 des W3C. Der Query-Standard stellt bei vielen Funktionen frei, wie viele diese zu implementieren sind. Im Folgenden finden Sie eine Liste, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Schemafähigkeit
Der Altova XQuery 1.0-Prozessor ist **schemafähig.**

Kodierung
Die UTF-8 und die UTF-16 Zeichen-Kodierungen werden unterstützt.

Namespaces
Die folgenden Namespace-URIs und die damit verknüpften Bindings sind vordefiniert.

Namespace-Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema-Instanz	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Vordefinierte Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/xquery-local-functions`

© 2014 Altova GmbH

Beachten Sie bitte die folgenden Punkte:

- Der Altova XQuery 1.0-Prozessor ist so konfiguriert, dass die oben aufgelisteten Präfixe an die entsprechenden Namespaces gebunden sind.
- Da der oben angeführte Namespace für vordefinierte Funktionen der Standard-Funktions-Namespace in XQuery ist, muss beim Aufruf von vordefinierten Funktionen das Präfix `fn:` nicht verwendet werden (`string("Hello")` ruft z.B. die Funktion `fn:string` auf). Das Präfix `fn:` kann jedoch verwendet werden, um eine vordefinierte Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Standard-Funktions-Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiele: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix verwenden möchten, muss dieses im Abfrageprolog für die Namespaces explizit deklariert werden. (Beispiel `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- Beachten Sie, dass die Datentypen `untypedAtomic`, `dayTimeDuration` und `yearMonthDuration` mit den Candidate Recommendations vom 23 January 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben wurden, d.h. `xs:yearMonthDuration`.

Wenn Namespaces für Funktionen, Typ-Konstruktoren, Node Tests usw. falsch zugewiesen wurden, wird ein Fehler ausgegeben. Beachten Sie jedoch, dass einige Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` ist definiert.) Das Namespace-Präfix legt fest, ob die Funktion oder der Typ-Konstruktor verwendet wird.

XML-Quelldokument und Validierung
XML-Dokumente, die bei der Ausführung eines XQuery-Dokuments mit dem Altova XQuery 1.0-Prozessor verwendet werden, müssen wohlgeformt sein. Sie müssen jedoch nicht gemäß einem XML-Schema gültig sein. Wenn die Datei nicht gültig ist, wird die ungültige Datei ohne Schemainformationen geladen. Wenn die XML-Datei mit einem externen Schema verknüpft ist und gemäß diesem Schema gültig ist, werden für die XML-Daten nachträglich Validierungsinformationen generiert und für die Auswertung der Abfrage verwendet.

Statische und dynamische Typ-Überprüfung
In der statischen Analysephase werden Aspekte der Abfrage überprüft wie z.B. die Syntax, ob externe Referenzen (z.B. für Module) vorhanden sind, ob aufgerufene Funktionen und Variablen definiert sind, usw. Wenn in dieser Phase ein Fehler gefunden wird, wird eine Meldung ausgegeben und die Ausführung wird gestoppt.

Die dynamische Typ-Überprüfung wird in Laufzeit durchgeführt, während die Abfrage ausgeführt

wird. Wenn ein Typ mit den Anforderungen einer Operation nicht kompatibel ist, wird ein Fehler ausgegeben. So gibt z.B. der Ausdruck `xs:string("1") + 1` einen Fehler zurück, weil die Operation "Addition" nicht an einem Operanden vom Typ `xs:string` ausgeführt werden kann.

Bibliotheksmodule
Bibliotheksmodule dienen zum Speichern von Funktionen und Variablen, damit diese wiederverwendet werden können. Der Altova XQuery 1.0-Prozessor unterstützt Module, die in einer **einzigen externen XQuery-Datei** gespeichert sind. Eine solche Moduldatei muss im Prolog eine `module`-Deklaration enthalten, in der ein Target Namespace zugewiesen wird. Hier ein Beispielmodul:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

Alle im Modul deklarierten Funktionen und Variablen gehören zu dem mit dem Modul verknüpften Namespace. Das Modul wird durch Import in eine XQuery-Datei mittels der `import module`-Anweisung im Abfrageprolog verwendet. Die `import module`-Anweisung importiert nur Funktionen und Variablen, die direkt in der Bibliotheksmodul-Datei deklariert sind:

```
import module namespace modlib = "urn:module-library" at
   "modulefilename.xq";
if    ($modlib:company = "Altova")
then   modlib:webaddress()
else   error("No match found.")
```

Externe Funktionen
Externe Funktionen, d.h. diejenigen Funktionen, die das Schlüsselwort `external` verwenden, werden nicht unterstützt:

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations
Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Andere unterstützte Collations sind die hier aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie ihre in der Liste der unterstützten Collations angeführte URI an. String-Vergleiche, wie die Funktionen `fn:max` und `fn:min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Präzision von numerischen Typen

 © 2014 Altova GmbH

- Der Datentyp `xs:integer` hat eine beliebige Präzision, d.h. er kann beliebig viele Stellen haben.
- Der Datentyp `xs:decimal` kann nach dem Dezimalpunkt maximal 20 Stellen haben.
- Die Datentypen `xs:float` und `xs:double` sind auf 15 Stellen beschränkt.

Unterstützung für XQuery-Anweisungen
Die `Pragma` Anweisung wird nicht unterstützt. Gegebenenfalls wird sie ignoriert und der Fallback-Ausdruck wird evaluiert.

8.5 XQuery 3.0

Der XQuery 3.0-Prozessor von RaptorXML Server entspricht der XQuery 3.0 Proposed Recommendation vom 8. April 2014 des World Wide Web Consortium (W3C) und unterstützt XPath- und XQuery-Funktionen 3.0.

Die implementierungsspezifischen Eigenschaften sind dieselben wie für XQuery 1.0.

© 2014 Altova GmbH

Kapitel 9

XSLT- und XPath/XQuery-Funktionen

9 XSLT- und XPath/XQuery-Funktionen

Dieser Abschnitt enthält eine Liste von Altova-Erweiterungsfunktionen und anderen Erweiterungsfunktionen, die in XPath und/oder XQuery-Ausdrücken verwendet werden können. Altova-Erweiterungsfunktionen können mit dem XSLT- und XQuery-Prozessor von Altova verwendet werden und bieten zusätzliche Funktionalitäten zu den in den W3C-Standards definierten Funktionsbibliotheken.

Allgemeine Punkte
Beachten Sie bitte die folgenden allgemeinen Punkte:

- Funktionen aus den in den W3C-Spezifikationen definierten core-Funktionsbibliotheken können ohne Präfix aufgerufen werden, da der XSLT- und XQuery-Prozessor Funktionen, die kein Präfix haben, als Funktionen des Standard-Funktions-Namespace liest. Der Standard-Funktions-Namespace ist derjenige, der in den XPath/XQuery-Funktionsspezifikationen definiert ist `http://www.w3.org/2005/xpath-functions`. Wenn dieser Namespace in einem XSLT- oder XQuery-Dokument explizit deklariert ist, kann das in der Namespace-Deklaration definierte Präfix optional auch in Funktionsnamen verwendet werden.
- Grundsätzlich gilt: Wenn bei einer Funktion eine Sequenz von einem Datenelement als Argument erwartet wird und eine Sequenz von mehr als einem Datenelement gesendet wird, wird ein Fehler zurückgegeben.
- Alle String-Vergleiche werden unter Verwendung der Unicode Codepoint Collation ausgeführt.
- Ergebnisse, bei denen es sich um QNames handelt, werden in der Form `[prefix:] localname` serialisiert.

Präzision von xs:decimal
Die Präzision bezieht sich auf die Anzahl der Stellen in einer Zahl. Laut Spezifikation sind mindestens 18 Stellen erforderlich. Bei Divisionen, bei denen ein Ergebnis vom Typ `xs:decimal` erzeugt wird, beträgt die Präzision 19 Kommastellen ohne Runden.

Implizite Zeitzone
Beim Vergleich zweier `date`, `time`, oder `dateTime` Werte muss die Zeitzone der verglichenen Werte bekannt sein. Wenn die Zeitzone in einem solchen Wert nicht explizit angegeben ist, wird die implizite Zeitzone verwendet. Als implizite Zeitzone wird die der Systemuhr verwendet. Der Wert kann mit Hilfe der Funktion `implicit-timezone()` überprüft werden.

Collations
Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Andere unterstützte Collations sind die unten aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie ihre in der Liste der unterstützten Collations (Tabelle unten) angeführte URI an. String-Vergleiche, wie die Funktionen `max` und `min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

 © 2014 Altova GmbH

Sprache	URIs
da: Dänisch	da_DK
de: Deutsch	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: Englisch	en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW
es: Spanisch	es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE
fr: Französisch	fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG
it: Italienisch	it_CH, it_IT
ja: Japanisch	ja_JP
nb: Norwegisch (Bokmal)	nb_NO
nl: Holländisch	nl_AW, nl_BE, nl_NL
nn: Norwegisch (Nynorsk)	nn_NO
pt: Portugiesisch	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Russisch	ru_MD, ru_RU, ru_UA
sv: Schwedisch	sv_FI, sv_SE

Namespace-Achse
Die Namespace-Achse wird in XPath 2.0 nicht mehr verwendet, wird aber weiterhin unterstützt. Um Namespace-Informationen mit XPath 2.0-Mechanismen aufzurufen, verwenden Sie die Funktionen `in-scope-prefixes()`, `namespace-uri()` und `namespace-uri-for-prefix()`.

9.1 Altova-Erweiterungsfunktionen

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath, XQuery und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

Die in der "XPath/XQuery Functions"-Spezifikation des W3C definierten Funktionen können (i) in einem XSLT-Kontext in XPath-Ausdrücken und (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden. In dieser Dokumentation sind die Funktionen, die im Zusammenhang mit XPath in XSLT verwendet werden können, mit einem **XP**-Symbol und Funktionen, die im Zusammenhang mit XQuery verwendet werden können, mit einem **XQ**-Symbol markiert; sie fungieren als XQuery-Funktionen. In den XSLT-Spezifikationen des W3C (nicht in den "XPath/XQuery Functions"-Spezifikationen) sind außerdem Funktionen definiert, die in XSLT-Dokumenten in XPath-Ausdrücken verwendet werden können. Diese Funktionen sind mit dem Symbol **XSLT** gekennzeichnet und werden als XSLT-Funktionen bezeichnet. In welcher XPath/XQuery- und XSLT-Version eine Funktion verwendet werden kann, wird in der Beschreibung der Funktion (*siehe Symbole unten)* angegeben. Funktionen aus der XPath/XQuery- und XSLT-Funktionsbibliothek werden ohne Präfix aufgelistet. Erweiterungsfunktionen aus anderen Bibliotheken wie z.B. Altova-Erweiterungsfunktionen werden mit einem Präfix angegeben.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XP1** **XP2** **XP3**
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XSLT1** **XSLT2** **XSLT3**
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	**XQ1** **XQ3**

XSLT-Funktionen
XSLT-Funktionen können in XPath-Ausdrücken nur im XSLT-Kontext verwendet werden (ähnlich wie die XSLT 2.0-Funktionen `current-group()` oder `key()`). Diese Funktionen sind nicht für Nicht-XSLT-Kontext gedachte und funktionieren in einem solchen Kontext (z.B. in einem XQuery-Kontext) nicht. Beachten Sie, dass XSLT-Funktionen für XBRL nur mit Altova Produkteditionen verwendet werden können, die XBRL unterstützen.

XPath/XQuery-Funktionen
XPath/XQuery-Funktionen (allgemeine, date/time und string) können sowohl in XPath-Ausdrücken im XSLT-Kontext als auch in XQuery-Ausdrücken verwendet werden.

Diagrammfunktionen (nur Enterprise- und Server-Editionen)
Altova-Erweiterungsfunktionen für Diagramme werden nur in der Enterprise und Server Edition von Altova-Produkten unterstützt und ermöglichen die Generierung von Diagrammen anhand von XML-Daten.

 © 2014 Altova GmbH

Barcode-Funktionen

Mit Hilfe der Altova Barcode-Funktionen können Barcodes generiert und in mittels XSLT Stylesheets generierte Ausgabedokumente eingefügt werden.

9.1.1 XSLT-Funktionen

XSLT-Erweiterungsfunktionen können in XPath-Ausdrücken in einem XSLT-Kontext verwendet werden. In einem Nicht-XSLT-Kontext (z.B. in einem XQuery-Kontext) funktionieren sie nicht.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache
Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath, XQuery und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3

Standardfunktionen

▼ **distinct-nodes [altova:]**

altova:distinct-nodes(*node()**)** als **node()*** XSLT1 XSLT2 XSLT3
Erhält eine Gruppe von einem oder mehreren Nodes als Input und gibt dieselbe Gruppe ohne Nodes mit doppelt vorhandenen Werten zurück. Der Vergleich wird mittels der XPath/XQuery-Funktion `fn:deep-equal` durchgeführt.

⊟ *Beispiele*

- **altova:distinct-nodes**(country) gibt alle Child `country` Nodes ohne diejenigen mit doppelt vorhandenen Werten zurück.

▼ **evaluate [altova:]**

altova:evaluate(XPathExpression *as xs:string***[, ValueOf$p1, ... ValueOf$pN])**
XSLT1 XSLT2 XSLT3
Erhält einen XPath-Ausdruck als obligatorisches Argument, der als String übergeben wird, und gibt das Resultat des ausgewerteten Ausdrucks zurück. Beispiel: **altova:evaluate('//Name[1]')** gibt den Inhalt des ersten `Name` Elements im Dokument zurück. Beachten Sie, dass der Ausdruck `//Name[1]` durch Einschließen in einfache Anführungszeichen als String übergeben wird.

Die Funktion `altova:evaluate` kann zusätzliche (optionale) Argumente erhalten. Diese Argumente sind die Werte der einzelnen im Geltungsbereich befindlichen Variablen und haben die Namen `p1`, `p2`, `p3`... `pN`. Beachten Sie zur Verwendung die folgenden Punkte: (i) Die Variablennamen müssen die Form `pX` haben, wobei `X` eine Ganzzahl ist; (ii) die Argumente der Funktion `altova:evaluate` (*siehe Signatur oben*) liefern vom zweiten Argument an die Werte der Variablen, wobei die Reihenfolge der Argumente der numerisch geordneten Variablensequenz entspricht: `p1` bis `pN`. Das zweite Argument wird der Wert der Variablen `p1`, das dritte Argument der der Variablen `p2`, usw.; (iii) Die Werte der Variablen müssen vom Typ `item*` sein

 © 2014 Altova GmbH

Beispiel

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

gibt aus `"hi 20 10"`

Beachten Sie im obigen Beispiel folgende Punkte:

- Das zweite Argument des Ausdrucks `altova:evaluate` ist der der Variablen `$p1` zugewiesene Wert, das dritte Argument ist das der Variablen `$p2` zugewiesene usw.
- Beachten Sie, dass das vierte Argument der Funktion ein String-Wert ist. Als String-Wert wird dieser innerhalb von Anführungszeichen gesetzt.
- Das `select` Attribut des Elements `xs:variable` liefert den XPath-Ausdruck. Da dieser Ausdruck den Typ `xs:string`, haben muss, wird er in einfache Anführungszeichen gesetzt.

Weitere Beispiele für die Verwendung der Variablen

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  Gibt den Wert des ersten `Name` *Elements zurück.*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Gibt "`//Name[1]`*" aus*

Die `altova:evaluate()` Erweiterungsfunktion ist in Situationen nützlich, in denen ein XPath-Ausdruck im XSLT-Stylesheet einen oder mehrere Teile enthält, die dynamisch ausgewertet werden müssen. Angenommen ein Benutzer gibt seinen Request für das Sortierkriterium ein und das Sortierkriterium ist im Attribut `UserReq/@sortkey` gespeichert. Im Stylesheet könnten Sie den folgenden Ausdruck haben: `<xsl:sort select="altova:evaluate(../UserReq/@sortkey)" order="ascending"/>` Die `altova:evaluate()` Funktion liest das `sortkey` Attribut des `UserReq` Child-Elements des Parent des Kontext-Node. Angenommen der Wert des `sortkey` Attributs ist `Price`, dann wird von der `altova:evaluate()` Funktion `Price` zurückgegeben und wird zum Wert des `select` Attributs: `<xsl:sort select="Price" order="ascending"/>`. Wenn diese `sort` Anweisung im Kontext eines Elements namens `Order` vorkommt, dann werden die `Order` Elemente nach den Werten Ihrer `Price` Children sortiert. Alternativ dazu, wenn der Wert von `@sortkey` z.B. `Date` ist, werden die `Order` Elemente nach den Werten ihrer `Date` Children sortiert. Das Sortierkriterium für `Order` wird also zur Laufzeit aus dem `sortkey` Attribut ausgewählt. Diese hätte man mit einem Ausdruck wie dem folgenden nicht bewerkstelligen können: `<xsl:sort select="../UserReq/@sortkey" order="ascending"/>`. Im oben gezeigten Beispiel wäre das Sortierkriterium das `sortkey` Attribut selbst, nicht `Price` oder `Date` (oder jeder beliebige andere Inhalt von `sortkey`)

Hinweis:
Der statische Kontext enthält Namespaces, Typen und Funktionen - aber keine Variablen - aus der aufrufenden Umgebung. Die Basis-URI und der Standard-Namespace werden vererbt.

Weitere Beispiele

- Statische Variablen: `<xsl:value-of select="$i3, $i2, $i1" />`
  Gibt die Werte von drei Variablen aus.

- Dynamischer XPath-Ausdruck mit dynamischen Variablen:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  Gibt "30 20 10" aus

- Dynamischer XPath-Ausdruck ohne dynamische Variable:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  Gibt einen Fehler aus.: Es wurde keine Variable für `$p3` *definiert.*

encode-for-rtf [altova:]

altova:encode-for-rtf(input *als xs:string*, **preserveallwhitespace** *als xs:boolean*, **preservenewlines** *als xs:boolean*) als **xs:string** XSLT2 XSLT3

Konvertiert den Input-String in Code für RTF. Whitespaces und neue Zeilen werden gemäß dem für die entsprechenden Parameter definierten Booleschen Wert beibehalten.

[Nach oben]

XBRL-Funktionen

Altova XBRL-Funktionen können nur mit Editionen von Altova-Produkten verwendet werden, die XBRL unterstützen.

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** als **node()*** XSLT2 XSLT3

Erhält einen Node als Input-Argument und gibt die durch den Input-Node referenzierte Gruppe der XBRL-Fußnoten-Nodes zurück.

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName*, *xs:string***)** als **node()*** XSLT2 XSLT3

Erhält zwei Input-Argumente: einen Node-Namen und den Pfad der Taxonomiedatei, die den Node enthält. Die Funktion gibt die XBRL Labels zurück, die mit dem Input-Node verknüpft sind.

[Nach oben]

 © 2014 Altova GmbH

9.1.2 XPath/XQuery-Funktionen: Datum und Uhrzeit

Die Datums- und Uhrzeit-Erweiterungsfunktionen von Altova können im Zusammenhang mit XPath- und XQuery-Ausdrücken verwendet werden und bieten zusätzliche Funktionalitäten für die Verarbeitung von Daten, die in Form von XML-Schema-Datums- und Uhrzeit-Datentypen zur Verfügung stehen. Diese Funktionen können mit dem **XPath 3.0-** und **XQuery 3.0-Prozessor** von Altova verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache
Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath, XQuery und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3

▼ Nach Funktionalität gruppiert

- Hinzufügen einer Zeitdauer zu xs:dateTime und Rückgabe von xs:dateTime
- Hinzufügen einer Zeitdauer zu xs:date und Rückgabe von xs:date
- Hinzufügen einer Zeitdauer zu xs:time und Rückgabe von xs:time
- Entfernen der Zeitzone aus Funktionen, die das aktuelle Datum/die aktuelle Uhrzeit generieren
- Rückgabe des Wochentags anhand des Datums als Ganzzahl
- Rückgabe eines Wochentags als Ganzzahl anhand eines Datums
- Erstellen des Datums, der Uhrzeit oder des Zeitdauertyps anhand der lexikalischen Komponenten der einzelnen Typen
- Konstruieren des Typs Datum, Datum und Uhrzeit oder Uhrzeit anhand eines String Input
- Funktionen zur Berechnung des Alters

▼ Datums- und Uhrzeitfunktionen in alphabetischer Reihenfolge:

altova:add-days-to-date
altova:add-days-to-dateTime
altova:add-hours-to-dateTime
altova:add-hours-to-time
altova:add-minutes-to-dateTime
altova:add-minutes-to-time
altova:add-months-to-date
altova:add-months-to-dateTime
altova:add-seconds-to-dateTime
altova:add-seconds-to-time
altova:add-years-to-date
altova:add-years-to-dateTime

altova:age
altova:age-details
altova:build-date
altova:build-duration
altova:build-time
altova:current-dateTime-no-TZ
altova:current-date-no-TZ
altova:current-time-no-TZ
altova:parse-date
altova:parse-dateTime
altova:parse-time
altova:weekday-from-date
altova:weekday-from-dateTime
altova:weeknumber-from-date
altova:weeknumber-from-dateTime

[Nach oben]

Hinzufügen einer Zeitdauer zu `xs:dateTime` XP3 XQ3

Mit diesen Funktionen werden Zeitdauerwerte zu **`xs:dateTime`** hinzugefügt, bevor **`xs:dateTime`** zurückgegeben wird. Der Typ `xs:dateTime` hat das Format `JJJJ-MM-TTZhh:mm:ss.sss`. Es handelt sich hierbei um eine Verkettung des `xs:date` und `xs:time` Formats, getrennt durch den Buchstaben `Z`. Ein Zeitzonensuffix`+01:00 (z.B.)` ist optional.

add-years-to-dateTime [altova:]

`altova:add-years-to-dateTime(DateTime` *`als xs:dateTime`*`, **Years** als xs:integer)` als **`xs:dateTime`** XP3 XQ3

Fügt eine Zeitdauer in Jahren zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Parameter angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **`altova:add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` gibt `2024-01-15T14:00:00` zurück
- **`altova:add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -4)` gibt `2010-01-15T14:00:00` zurück

add-months-to-dateTime [altova:]

`altova:add-months-to-dateTime(DateTime` *`als xs:dateTime`*`, **Months** als xs:integer)` als **`xs:dateTime`** XP3 XQ3

Fügt eine Zeitdauer in Monaten zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **`altova:add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` gibt `2014-11-15T14:00:00` zurück
- **`altova:add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -2)` gibt `2013-11-15T14:00:00` zurück

 © 2014 Altova GmbH

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *als xs:dateTime*, **Days** *als xs:integer*) als **xs:dateTime** **XP3** **XQ3**

Fügt eine Zeitdauer in Tagen zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2014-01-25T14:00:00 zurück
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) gibt 2014-01-07T14:00:00 zurück

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *als xs:dateTime*, **Hours** *als xs:integer*) als **xs:dateTime** **XP3** **XQ3**

Fügt eine Zeitdauer in Stunden zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) gibt 2014-01-15T23:00:00 zurück
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) gibt 2014-01-15T05:00:00 zurück

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *als xs:dateTime*, **Minutes** *als xs:integer*) als **xs:dateTime** **XP3** **XQ3**

Fügt eine Zeitdauer in Minuten zu einem `xs:dateTime` Wert (*siehe Beispiele unten)* hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) gibt 2014-01-15T14:55:00 zurück
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) gibt 2014-01-15T14:05:00 zurück

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *als xs:dateTime*, **Seconds** *als xs:integer*) als **xs:dateTime** **XP3** **XQ3**

Fügt eine Zeitdauer in Sekunden zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten

Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) gibt 2014-01-15T14:00:30 zurück
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) gibt 2014-01-15T14:00:05 zurück

[Nach oben]

Hinzufügen einer Zeitdauer zu `xs:date` XP3 XQ3

Mit diesen Funktionen werden Zeitdauerwerte zu **xs:date** hinzugefügt, bevor **xs:date** zurückgegeben wird. Der Typ `xs:date` hat das Format `JJJJ-MM-TT`.

add-years-to-date [altova:]

altova:add-years-to-date(Date *als xs:date*, **Years** *als xs:integer***)** als **xs:date** XP3 XQ3

Fügt eine Zeitdauer in Jahren zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) gibt 2024-01-15 zurück
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) gibt 2010-01-15 zurück

add-months-to-date [altova:]

altova:add-months-to-date(Date *als xs:date*, **Months** *als xs:integer***)** als **xs:date** XP3 XQ3

Fügt eine Zeitdauer in Monaten zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) gibt 2014-11-15 zurück
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) gibt 2013-11-15 zurück

add-days-to-date [altova:]

altova:add-days-to-date(Date *als xs:date*, **Days** *als xs:integer***)** als **xs:date** XP3 XQ3

Fügt eine Zeitdauer in Tagen zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) gibt 2014-01-25 zurück
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) gibt 2014-01-07 zurück

[Nach oben]

 © 2014 Altova GmbH

Hinzufügen einer Zeitdauer zu `xs:time` XP3 XQ3

Diese Funktionen fügen einen Zeitdauerwert zu `xs:time` hinzu und geben `xs:time` zurück. Der Typ `xs:time` entspricht in seiner lexikalischen Form `hh:mm:ss.sss`. Eine optionale Zeitzone kann angehängt werden. Der Buchstabe `Z` steht für Coordinated Universal Time (UTC). Alle anderen Zeitzonen werden in Form des Zeitunterschieds zur UTC im Format `+hh:mm`, oder `-hh:mm` dargestellt. Wenn kein Wert für die Zeitzone vorhanden ist, wird sie als unbekannt und nicht als UTC angenommen.

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *als xs:time*, **Hours** *als xs:integer*) als **xs:time** XP3 XQ3

Fügt eine Zeitdauer in Stunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`.

Beispiele

- `altova:add-hours-to-time(xs:time("11:00:00"), 10)` gibt `21:00:00` zurück
- `altova:add-hours-to-time(xs:time("11:00:00"), -7)` gibt `04:00:00` zurück

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *als xs:time*, **Minutes** *als xs:integer*) als **xs:time** XP3 XQ3

Fügt eine Zeitdauer in Minuten zu einem `xs:time` Wert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`.

Beispiele

- `altova:add-minutes-to-time(xs:time("14:10:00"), 45)` gibt `14:55:00` zurück
- `altova:add-minutes-to-time(xs:time("14:10:00"), -5)` gibt `14:05:00` zurück

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *als xs:integer*) als **xs:time** XP3 XQ3

Fügt eine Zeitdauer in Sekunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`. Die Seconds Komponenten kann sich im Bereich von `0` bis `59.999` befinden.

Beispiele

- `altova:add-seconds-to-time(xs:time("14:00:00"), 20)` gibt `14:00:20` zurück
- `altova:add-seconds-to-time(xs:time("14:00:00"), 20.895)` gibt `14:00:20.895` zurück

[Nach oben]

Entfernen der Zeitzone aus date/time-Datentypen XP3 XQ3

Diese Funktionen entfernen die Zeitzone aus den aktuellen **xs:dateTime**, **xs:date** bzw. **xs:time** Werten. Beachten Sie, dass im Unterschied zu `xs:dateTime` bei `xs:dateTimeStamp` die Zeitzone erforderlich ist (während sie im ersteren Fall optional ist). Das Format eines `xs:dateTimeStamp` Werts lautet daher: `JJJJ-MM-TTZhh:mm:ss.sss±hh:mm.` oder `JJJJ-MM-TTZhh:mm:ss.sssZ`. Wenn das Datum und die Uhrzeit von der Systemuhr als `xs:dateTimeStamp` ausgelesen wird, können Sie die Zeitzone, falls erforderlich, mit der Funktion `current-dateTime-no-TZ()` entfernen.

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() als **xs:dateTime** XP3 XQ3

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-dateTime()` Wert (welcher das aktuelle Datum und die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen `xs:dateTime` Wert zurück.

Beispiele

Wenn der aktuelle Datums- und Uhrzeitwert **2014-01-15T14:00:00+01:00** lautet:

- **altova:current-dateTime-no-TZ()** gibt `2014-01-15T14:00:00` zurück

current-date-no-TZ [altova:]

altova:current-date-no-TZ() als **xs:date** XP3 XQ3

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-date()` Wert (welcher das aktuelle Datum laut Systemuhr ist) und gibt einen `xs:date` Wert zurück.

Beispiele

Wenn das aktuelle Datum **2014-01-15+01:00** lautet:

- **altova:current-date-no-TZ()** gibt `2014-01-15` zurück

current-time-no-TZ [altova:]

altova:current-time-no-TZ() als **xs:time** XP3 XQ3

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-time()` Wert (welcher die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen `xs:time` Wert zurück.

Beispiele

Wenn der aktuelle Uhrzeitwert **14:00:00+01:00** lautet:

- **altova:current-time-no-TZ()** gibt `14:00:00` zurück

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date***)** als **xs:date** XP3 XQ3

Diese Funktion verwendet ein `xs:date` Argument, entfernt den Zeitzonenteil daraus und gibt einen `xs:date` Wert zurück. Beachten Sie, dass das Datum nicht geändert wird..

Beispiele

- **altova:date-no-TZ**`(xs:date("2014-01-15+01:00"))` gibt `2014-01-15` zurück

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *als xs:dateTime***)** als **xs:dateTime** XP3 XQ3

Diese Funktion verwendet ein `xs:dateTime` Argument, entfernt den Zeitzonenteil daraus und gibt einen `xs:dateTime` Wert zurück. Beachten Sie, dass weder Datum noch Uhrzeit geändert werden.

 © 2014 Altova GmbH

Beispiele

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) gibt 2014-01-15T14:00:00 zurück

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *als xs:time***)** als **xs:time** XP3 XQ3

Diese Funktion verwendet ein xs:time Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:time Wert zurück. Beachten Sie, dass die Uhrzeit nicht geändert wird.

Beispiele

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) gibt 14:00:00 zurück

[Nach oben]

Rückgabe des Wochentages anhand von xs:dateTime oder xs:date XP3 XQ3

Diese Funktionen geben anhand des xs:dateTime oder xs:date Werts den Wochentag in Form einer Ganzzahl zurück. Die Tage der Woche sind (im amerikanischen Format) von 1 bis 7 nummeriert, wobei Sonntag=1. Im europäischen Format beginnt die Woche am Montag (=1). Das amerikanische Format, in dem Sonntag=1, kann mittels der Ganzzahl 0 definiert werden, wenn das Format mittels einer Ganzzahl angegeben werden kann.

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *als xs:dateTime***)** als **xs:integer** XP3 XQ3

Erhält ein Datum mit einer Uhrzeit als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit Sonntag=1 nummeriert. Wenn das europäische Format benötigt wird (wo Montag=1), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten)*.

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) gibt 2 zurück, wobei 2 für Montag steht.

altova:weekday-from-dateTime(DateTime *als xs:dateTime*, **Format** *als xs:integer***)** als **xs:integer** XP3 XQ3

Erhält ein Datum mit einer Uhrzeit als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit Montag=1 nummeriert. Wenn das zweite (Integer)-Argument 0 ist, werden die Wochentage beginnend mit Sonntag=1 von 1 bis 7 nummeriert. Wenn das zweite Argument eine andere Ganzzahl als 0 ist, so ist Montag=1. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur).*

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) gibt 1, zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) gibt 1, zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) gibt 2, zurück, wobei 2 für Montag steht

weekday-from-date [altova:]

altova:weekday-from-date(Date *als xs:date*) als **xs:integer** XP3 XQ3

Erhält ein Datum als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Sonntag=1` nummeriert. Wenn das europäische Format benötigt wird (wo `Montag=1`), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten)*.

Beispiele

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) gibt `2` zurück, wobei `2` für Montag steht.

altova:weekday-from-date(Date *als xs:date*, **Format** *als xs:integer*) als **xs:integer** XP3 XQ3

Erhält ein Datum als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Montag=1` nummeriert. Wenn das zweite Argument (`Format`) `0` ist, werden die Wochentage beginnend mit `Sonntag=1` von `1` bis `7` nummeriert. Wenn das zweite Argument eine andere Ganzzahl als `0` ist, so ist `Montag=1`. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur).*

Beispiele

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) gibt `1`, zurück, wobei `1` für Montag steht
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) gibt `1`, zurück, wobei `1` für Montag steht
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) gibt `2`, zurück, wobei `2` für Montag steht

[Nach oben]

Rückgabe der Wochennummer anhand von `xs:dateTime` oder `xs:date` XP2 XQ1 XP3 XQ3

Diese Funktionen geben anhand von `xs:dateTime` oder `xs:date` die Wochennummer als Ganzzahl zurück. Die Wochennummer steht in den Kalenderformaten US, ISO/European und Islamic zur Verfügung. Die Wochennummerierung unterscheidet sich in diesen Kalenderformaten, da die Woche in diesen Formaten an unterschiedlichen Tagen beginnt (Im Format US am Sonntag, im Format ISO/European am Montag und im Format Islamic am Samstag).

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *als xs:date*, **Calendar** *als xs:integer*) als **xs:integer** XP2 XQ1 XP3 XQ3

Gibt die Wochennummer des bereitgestellten **`Date`** Arguments als Ganzzahl zurück. Das zweite Argument (**`Calendar`**) definiert das zu verwendende Kalendersystem.

Unterstützte **`Calendar`** Werte sind:

- **`0 = US-Kalender`** (Woche beginnt am Sonntag)
- **`1 = ISO-Standard, Europäischer Kalender`** (Woche beginnt am Montag)
- **`2 = Islamischer Kalender`** (Woche beginnt am Samstag)

Default is `0`.

 © 2014 Altova GmbH

Beispiele

- `altova:weeknumber-from-date(xs:date("2014-03-23"), 0)` gibt `13` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 1)` gibt `12` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 2)` gibt `13` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23") )` gibt `13` zurück

Der Tag des Datums in den obigen Beispielen (`2014-03-23`) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *als xs:dateTime*, **Calendar** *als xs:integer*) als **xs:integer** XP2 XQ1 XP3 XQ3

Gibt die Wochennummer des bereitgestellten **DateTime** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem. Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (*Woche beginnt am Sonntag*)
- **1 = ISO-Standard, Europäischer Kalender** (*Woche beginnt am Montag*)
- **2 = Islamischer Kalender** (*Woche beginnt am Samstag*)

Der Standardwert ist `0`.

Beispiele

- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0)` gibt `13` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1)` gibt `12` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2)` gibt `13` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00") )` gibt `13` zurück

Der Tag des Datums- und Uhrzeitwerts in den obigen Beispielen (`2014-03-23T00:00:00`) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

[Nach oben]

Erstellen des Datums-, Uhrzeit- oder Zeitdauer-Datentyps anhand der lexikalischen Komponenten der einzelnen Typen XP3 XQ3

Die Funktionen erhalten die lexikalischen Komponenten des `xs:date`, `xs:time` oder `xs:duration`-Datentyps als Input-Argumente und kombinieren diese zum entsprechenden Datentyp.

build-date [altova:]

altova:build-date(Year *als xs:integer*, **Month** *als xs:integer*, **Date** *als*

xs:integer) als **xs:date** XP3 XQ3

Das erste, zweite und dritte Argument steht für das Jahr, bzw. den Monat bzw. das Datum. Sie werden zu einem Wert vom Typ `xs:date` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Datumsteils befinden. So sollte z.B. das zweite Argument nicht größer als `12` sein.

Beispiele

- **altova:build-date**(2014, 2, 03) gibt 2014-02-03 zurück

build-time [altova:]

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:time** XP3 XQ3

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Sie werden zu einem Wert von Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. der zweite Parameter nicht größer als `59` sein. Um eine Zeitzone zum Wert hinzuzufügen, verwenden Sie die andere Signatur der Funktion (*siehe nächste Signatur)*.

Beispiele

- **altova:build-time**(23, 4, 57) gibt 23:04:57 zurück

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*, **TimeZone** *als xs:string*) als **xs:time** XP3 XQ3

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Das vierte Argument ist ein String, der den Zeitzonenteil des Werts liefert. Die vier Argumente werden zu einem Wert vom Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. das zweite Argument (Minuten) nicht größer als `59` sein.

Beispiele

- **altova:build-time**(23, 4, 57, '+1') gibt 23:04:57+01:00 zurück

build-duration [altova:]

altova:build-duration(Years *als xs:integer*, **Months** *als xs:integer*) als **xs:yearMonthDuration** XP3 XQ3

Setzt aus zwei Argumenten einen Wert vom Typ `xs:yearMonthDuration` zusammen. Das erste Argument liefert den Jahr-Teil des Zeitdauerwerts, während das zweite Argument den Monat-Teil liefert. Wenn der zweite Parameter (Monate) größer oder gleich `12` ist, so wird die Ganzzahl durch 12 dividiert. Der Quotient wird zum ersten Argument hinzugefügt, um den Jahr-Teil des Zeitdauerwerts zu liefern, während der Rest (der Division) den Monat-Teil liefert. Eine Beschreibung zur Erstellung einer Zeitdauer vom Typ `xs:dayTimeDuration` finden Sie in der nächsten Signatur.

Beispiele

- **altova:build-duration**(2, 10) gibt P2Y10M zurück
- **altova:build-duration**(14, 27) gibt P16Y3M zurück
- **altova:build-duration**(2, 24) gibt P4Y zurück

altova:build-duration(Days *als xs:integer*, **Hours** *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:dayTimeDuration** XP3 XQ3

 © 2014 Altova GmbH

Kombiniert vier Argumente zu einem Wert vom Typ `xs:dayTimeDuration`. Das erste Argument liefert den Tage-Teil, das zweite die Stunden, das dritte die Minuten und das vierte die Sekunden des Zeitdauerwerts. Die einzelnen Uhrzeitparameter werden in den entsprechenden Wert für die nächsthöhere Einheit konvertiert und das Ergebnis wird zur Berechnung der Gesamtdauer weitergegeben. So werden z.B. 72 Sekunden in `1M(inute) 12S(ekunden)` konvertiert. Dieser Wert wird zur Berechnung der Gesamtdauer weitergegeben. Um eine Zeitdauer vom Typ `xs:yearMonthDuration` zu berechnen, verwenden Sie die vorherige Signatur.

Beispiele

- **altova:build-duration**(2, 10, 3, 56) gibt P2DT10H3M56S zurück
- **altova:build-duration**(1, 0, 100, 0) gibt P1DT1H40M zurück
- **altova:build-duration**(1, 0, 0, 3600) gibt P1DT1H zurück

[Nach oben]

Konstruieren von Datum, Datum und Uhrzeit und Zeit-Datentypen anhand des String-Input XP2 XQ1 XP3 XQ3

Diese Funktionen erhalten Strings als Argumente und konstruieren anhand dieser die Datentypen `xs:date`, `xs:dateTime` oder `xs:time`. Der String wird anhand eines bereitgestellten Pattern-Arguments nach Komponenten des Datentyps analysiert.

parse-date [altova:]

altova:parse-date(**Date** *als xs:string*, **DatePattern** *als xs:string*) als **xs:date** XP2 XQ1 XP3 XQ3

Gibt den Input-String **Date** als **xs:date** Wert zurück. Das zweite Argument **DatePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DatePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr

Das Pattern in **DatePattern** muss mit dem Pattern in **Date** übereinstimmen. Da die Ausgabe vom Typ **xs:date** ist, hat sie immer das lexikalische Format **YYYY-MM-DD**.

Beispiele

- **altova:parse-date**(xs:string("06-03-2014"), "[D]-[M]-[Y]") gibt 2014-03-06 zurück
- **altova:parse-date**(xs:string("06-03-2014"), "[M]-[D]-[Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") gibt 2014-06-03 zurück

parse-dateTime [altova:]

altova:parse-dateTime(**DateTime** *als xs:string*, **DateTimePattern** *als xs:string*) als **xs:dateTime** XP2 XQ1 XP3 XQ3

Gibt den Input-String **DateTime** als **xs:dateTime** Wert zurück. Das zweite Argument **DateTimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DateTimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr
H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **DateTimePattern** muss mit dem Pattern in **DateTime** übereinstimmen. Da die Ausgabe vom Typ **xs:dateTime** ist, hat sie immer das lexikalische Format **YYYY-MM-DDTHH:mm:ss**.

Beispiele

- **altova:parse-dateTime**(xs:string("06-03-2014 13:56:24"), "[D]-[M]-[Y] [H]:[m]:[s]") gibt 2014-03-06T13:56:24 zurück
- **altova:parse-dateTime**("time=13:56:24; date=06-03-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") gibt 2014-03-06T13:56:24 zurück

parse-time [altova:]

altova:parse-time(**Time** *als xs:string*, **TimePattern** *als xs:string*) als **xs:time**
XP2 **XQ1** **XP3** **XQ3**

Gibt den Input-String **Time** als **xs:time** Wert zurück. Das zweite Argument **TimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **TimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **TimePattern** muss mit dem Pattern in **Time** übereinstimmen. Da die Ausgabe vom Typ **xs:Time** ist, hat sie immer das lexikalische Format **HH:mm:ss**.

Beispiele

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") gibt 13:56:24 zurück
- **altova:parse-time**("13-56-24", "[H]-[m]") gibt 13:56:00 zurück
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") gibt 13:56:24 zurück
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") gibt 13:56:24 zurück

[Nach oben]

 © 2014 Altova GmbH

Funktionen zur Berechnung des Alters XP3 XQ3

Diese Funktionen geben das Alter berechnet (i) anhand von einem Input-Argument und dem aktuellen Datum oder (ii) anhand zweier Input-Argumentdaten zurück. Die Funktion **altova:age** gibt das Alter in Jahren zurück, die Funktion **altova:age-details** gibt das Alter als Sequenz von drei Ganzzahlen zurück, die die Jahre, Monate und Tage des Alters angeben.

age [altova:]

altova:age(StartDate *als xs:date***)** als **xs:integer** XP3 XQ3

Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das Argument gelieferten Startdatums endend mit dem aktuellen Datum (laut Systemuhr). Wenn das Input-Argument eines Datums größer oder gleich einem Jahr in der Zukunft ist, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- `altova:age(xs:date("2013-01-15"))` gibt `1` zurück
- `altova:age(xs:date("2013-01-16"))` gibt `0` zurück
- `altova:age(xs:date("2015-01-15"))` gibt `-1` zurück
- `altova:age(xs:date("2015-01-14"))` gibt `0` zurück

altova:age(StartDate *als xs:date*, **EndDate** *als xs:date***)** als **xs:integer** XP3 XQ3

Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das erste Argument gelieferten Startdatums endend mit dem als zweites Datum gelieferten Enddatum. Wenn das erste Argument ein Jahr oder mehr nach dem zweiten Argument liegt, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- `altova:age(xs:date("2000-01-15"), xs:date("2010-01-15"))` gibt `10` zurück
- `altova:age(xs:date("2000-01-15"), current-date())` gibt `14` zurück, wenn das aktuelle Datum `2014-01-15` ist
- `altova:age(xs:date("2014-01-15"), xs:date("2010-01-15"))` gibt `-4` zurück

age-details [altova:]

altova:age-details(InputDate *als xs:date***)** als **(xs:integer)*** XP3 XQ3

Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen dem als Argument angegebenen Datum und dem aktuellen Datum (laut Systemuhr). Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Datumswerten (dem Input-Datum und dem aktuellen Datum) an. Das Input-Datum hat eventuell einen Wert, der vor oder nach dem aktuellen Datum liegt, doch wird dies nicht aus dem Vorzeichen der Rückgabewerte ersichtlich; die Rückgabewerte sind immer positiv.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- `altova:age-details(xs:date("2014-01-16"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2014-01-14"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2013-01-16"))` gibt `(1 0 1)` zurück
- `altova:age-details(current-date())` gibt `(0 0 0)` zurück

altova:age-details(Date-1 *als xs:date*, **Date-2** *als xs:date***)** als **(xs:integer)***

XP3 **XQ3**

Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen den beiden Argumentdaten. Die Summe der zurückgegebenen `years+months +days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Input-Datumswerten an. Es ist unerheblich, ob das frühere oder spätere Datum als erstes Argument angegeben wird. Die Rückgabewerte geben nicht an, ob das Input-Datum vor oder nach dem aktuellen Datum liegt. Die Rückgabewerte sind immer positiv.

Beispiele

- `altova:age-details(xs:date("2014-01-16"), xs:date("2014-01-15"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2014-01-15"), xs:date("2014-01-16"))` gibt `(0 0 1)` zurück

[Nach oben]

 © 2014 Altova GmbH

9.1.3 XPath/XQuery-Funktionen: String

Die folgenden XPath/XQuey-Erweiterungsfunktionen für Strings werden in der aktuellen Version Ihres Altova-Produkts unterstützt und können (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache
Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath, XQuery und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace**, `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3

---

camel-case [altova:]

altova:camel-case(InputString *als xs:string***)** als **xs:string** XP3 XQ3
Gibt den Input-String **InputString** in CamelCase zurück. Der String wird mit Hilfe der Regular Expression `'\s'` (welches ein Kürzel für das Leerzeichen ist) analysiert. Das erste Zeichen nach einem Leerzeichen oder einer Sequenz aufeinanderfolgender Leerzeichen, das kein Leerzeichen ist, wird mit einem Großbuchstaben geschrieben. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **altova:camel-case**("max") gibt Max zurück
- **altova:camel-case**("max max") gibt Max Max zurück
- **altova:camel-case**("file01.xml") gibt File01.xml zurück
- **altova:camel-case**("file01.xml file02.xml") gibt File01.xml File02.xml zurück
- **altova:camel-case**("file01.xml file02.xml") gibt File01.xml File02.xml zurück
- **altova:camel-case**("file01.xml -file02.xml") gibt File01.xml -file02.xml zurück

altova:camel-case(InputString *als xs:string*, **SplitChars** *als xs:string*, **IsRegex** *als xs:boolean***)** als **xs:string** XP3 XQ3
Konvertiert den Input-String **InputString** in CamelCase, indem anhand von **SplitChars** festgelegt wird, welche(s) Zeichen die nächste Konvertierung in Großbuchstaben auslöst. **SplitChars** wird als Regular Expression verwendet, wenn **IsRegex = true()** oder als einfache Zeichen, wenn **IsRegex = false()**. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **altova:camel-case**("setname getname", "set|get", true()) gibt setName getName zurück
- **altova:camel-case**("altova\documents\testcases", "\", false()) gibt Altova \Documents\Testcases zurück

char [altova:]

altova:char(Position *as xs:integer***)** als **xs:string** XP3 XQ3
Gibt einen String zurück, der das Zeichen an der durch das Argument Position definierten Position enthält. Dieses Zeichen wird durch Konvertierung des Werts des Kontextelements in xs:string ermittelt. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:char**(2) gibt 2 zurück
- **altova:char**(5) gibt A zurück
- **altova:char**(9) gibt den leeren String zurück.
- **altova:char**(-2) gibt den leeren String zurück.

altova:char(InputString *als xs:string*, **Position** *als xs:integer***)** als **xs:string** XP3 XQ3
Gibt einen String zurück, der das Zeichen enthält, das sich in dem als InputString Argument gelieferten String an der durch das Argument Position definierten Position befindet. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

Beispiele

- **altova:char**("2014-01-15", 5) gibt - zurück
- **altova:char**("USA", 1) gibt U zurück
- **altova:char**("USA", 1) gibt den leeren String zurück.
- **altova:char**("USA", -2) gibt den leeren String zurück.

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer***)** als **xs:string** XP3 XQ3
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch X-Number) des String enthält, der durch Konvertierung des Werts des Kontextelements in xs:string erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:first-chars**(2) gibt 12 zurück
- **altova:first-chars**(5) gibt 1234A zurück
- **altova:first-chars**(9) gibt 1234ABCD zurück

altova:first-chars(InputString *als xs:string*, **X-Number** *als xs:integer***)** als **xs:string** XP3 XQ3
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch X-Number) des String enthält, das als das Argument InputString angegeben ist.

Beispiele

- **altova:first-chars**("2014-01-15", 5) gibt 2014- zurück
- **altova:first-chars**("USA", 1) gibt U zurück

 © 2014 Altova GmbH

last-chars [altova:]

altova:last-chars(X-Number *als xs:integer***)** als **xs:string** XP3 XQ3
Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch X-Number) des String enthält, der durch Konvertierung des Werts des Kontextelements in xs:string erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:last-chars**(2) gibt CD zurück
- **altova:last-chars**(5) gibt 4ABCD zurück
- **altova:last-chars**(9) gibt 1234ABCD zurück

altova:last-chars(InputString *als xs:string*, **X-Number** *als xs:integer***)** als **xs:string** XP3 XQ3

Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch X-Number) des String enthält, das als das Argument InputString angegeben ist.

Beispiele

- **altova:last-chars**("2014-01-15", 5) gibt 01-15- zurück
- **altova:last-chars**("USA", 10) gibt USA zurück

pad-string-left [altova:]

altova:pad-string-left(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string***)** als **xs:string** XP3 XQ3
Das Argument PadCharacter ist ein einzelnes Zeichen, das links vom String, der als das Argument StringToPad geliefert wird, mit einem Auffüllzeichen versehen wird. Das Argument Repeats gibt an, wie oft das Auffüllzeichen links von StringToPad wiederholt werden soll.

Beispiele

- **altova:pad-string-left**('AP', 1, 'Z') gibt 'ZAP' zurück
- **altova:pad-string-left**('AP', 3, 'Z') gibt 'ZZZAP' zurück
- **altova:pad-string-left**('AP', 0, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 3, 'YZ') gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

pad-string-right [altova:]

altova:pad-string-right(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string***)** als **xs:string** XP3 XQ3
Das Argument PadCharacter ist ein einzelnes Zeichen, das rechts vom String, der als das Argument StringToPad geliefert wird, mit einem Auffüllzeichen versehen wird. Das Argument Repeats gibt an, wie oft das Auffüllzeichen rechts von StringToPad wiederholt werden soll.

Beispiele

- **altova:pad-string-right**('AP', 1, 'Z') gibt 'ZAP' zurück
- **altova:pad-string-right**('AP', 3, 'Z') gibt 'APZZZ' zurück
- **altova:pad-string-right**('AP', 0, 'Z') gibt 'AP' zurück
- **altova:pad-string-right**('AP', 3, 'YZ') gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

repeat-string [altova:]

altova:repeat-string(InputString *als xs:string*, **Repeats** *als xs:integer*) als **xs:string** **XP2 XQ1 XP3 XQ3**

Generiert einen String, der sich zusammensetzt aus dem ersten `InputString`-Argument, das die Anzahl der `Repeats` wiederholt wird.

Beispiele

- **altova:repeat-string**("Altova #", 3) gibt "Altova #Altova #Altova #" zurück

substring-after-last [altova:]

altova:substring-after-last(MainString *als xs:string*, **CheckString** *als xs:string*) als **xs:string** **XP3 XQ3**

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` nach `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring nach der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- **altova:substring-after-last**('ABCDEFGH', 'B') gibt 'CDEFGH' zurück
- **altova:substring-after-last**('ABCDEFGH', 'BC') gibt 'DEFGH'zurück
- **altova:substring-after-last**('ABCDEFGH', 'BD') gibt '' zurück
- **altova:substring-after-last**('ABCDEFGH', 'Z') gibt '' zurück
- **altova:substring-after-last**('ABCDEFGH', '') gibt 'ABCDEFGH' zurück
- **altova:substring-after-last**('ABCD-ABCD', 'B') gibt 'CD' zurück
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') gibt '' zurück

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*) as **xs:string** **XP3 XQ3**

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` vor `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring vor der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- **altova:substring-before-last**('ABCDEFGH', 'B') gibt 'A' zurück
- **altova:substring-before-last**('ABCDEFGH', 'BC') gibt 'A' zurück
- **altova:substring-before-last**('ABCDEFGH', 'BD') gibt '' zurück
- **altova:substring-before-last**('ABCDEFGH', 'Z') gibt '' zurück
- **altova:substring-before-last**('ABCDEFGH', '') gibt '' zurück
- **altova:substring-before-last**('ABCD-ABCD', 'B') gibt 'ABCD-A' zurück
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') gibt 'ABCD-ABCD-' zurück

 © 2014 Altova GmbH

substring-pos [altova:]

altova:substring-pos(StringToCheck *als xs:string*, **StringToFind** *als xs:string*) als **xs:integer** XP3 XQ3

Gibt die Zeichenposition der ersten Instanz von `StringToFind` im String `StringToCheck` zurück. Die Zeichenposition wird in Form einer Ganzzahl angegeben. Das erste Zeichen von `StringToCheck` hat die Position `1`. Wenn `StringToFind` in `StringToCheck` nicht vorkommt, wird die Ganzzahl `0` zurückgegeben. Um den String auf eine zweite oder eine weiter hinten folgende Instanz von `StringToCheck` zu überprüfen, verwenden Sie die nächste Signatur dieser Funktion.

Beispiele

- **altova:substring-pos**('Altova', 'to') gibt 3 zurück
- **altova:substring-pos**('Altova', 'tov') gibt 3 zurück
- **altova:substring-pos**('Altova', 'tv') gibt 0 zurück
- **altova:substring-pos**('AltovaAltova', 'to') gibt 3 zurück

altova:substring-pos(StringToCheck *als xs:string*, **StringToFind** *als xs:string*, **Integer** *als xs:integer*) als **xs:integer** XP3 XQ3

Gibt die Zeichenposition von `StringToFind` im String `StringToCheck` zurück. Die Suche nach `StringToFind` beginnt an der durch das Argument `Integer` angegebenen Zeichenposition; der Zeichen-Substring vor dieser Position wird nicht durchsucht. Die zurückgegebene Ganzzahl gibt jedoch die Position des gefundenen String innerhalb des *gesamten* String `StringToCheck` an. Diese Signatur dient dazu, die zweite oder eine weiter hinten folgende Position eines String zu finden, der mehrmals in `StringToCheck` vorkommt. Wenn `StringToFind` in `StringToCheck` nicht vorkommt, wird die Ganzzahl `0` zurückgegeben.

Beispiele

- **altova:substring-pos**('Altova', 'to', 1) gibt 3 zurück
- **altova:substring-pos**('Altova', 'to', 3) gibt 3 zurück
- **altova:substring-pos**('Altova', 'to', 4) gibt 0 zurück
- **altova:substring-pos**('Altova-Altova', 'to', 0) gibt 3 zurück
- **altova:substring-pos**('Altova-Altova', 'to', 4) gibt 10 zurück

trim-string [altova:]

altova:trim-string(InputString *als xs:string*) als **xs:string** XP3 XQ3

Diese Funktion verwendet ein `xs:string` Argument, entfernt alle voran- und nachgestellten Leerzeichen und gibt einen "getrimmten" `xs:string` zurück.

Beispiele

- **altova:trim-string**(" Hello World ")) gibt "Hello World" zurück
- **altova:trim-string**("Hello World ")) gibt "Hello World" zurück
- **altova:trim-string**(" Hello World")) gibt "Hello World" zurück
- **altova:trim-string**("Hello World")) gibt "Hello World" zurück
- **altova:trim-string**("Hello World")) gibt "Hello World" zurück

trim-string-left [altova:]

altova:trim-string-left(InputString *als xs:string*) als **xs:string** XP3 XQ3

Diese Funktion verwendet ein `xs:string` Argument, entfernt alle vorangestellten Leerzeichen und gibt einen "links getrimmten" `xs:string` zurück.

Beispiele

- **altova:trim-string-left**(" Hello World ")) gibt "Hello World " zurück
- **altova:trim-string-left**("Hello World ")) gibt "Hello World " zurück
- **altova:trim-string-left**(" Hello World")) gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World")) gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World")) gibt "Hello World" zurück

trim-string-right [altova:]

altova:trim-string-right(InputString *als xs:string*) als **xs:string** XP3 XQ3

Diese Funktion verwendet ein xs:string Argument, entfernt alle nachgestellten Leerzeichen und gibt einen "rechts getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-right**(" Hello World ")) gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World ")) gibt "Hello World" zurück
- **altova:trim-string-right**(" Hello World")) gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World")) gibt "Hello World" zurück
- **altova:trim-string-right**("Hello World")) gibt "Hello World" zurück

© 2014 Altova GmbH

9.1.4 XPath/XQuery-Funktionen: Diverse Funktionen

Die folgenden XPath/XQuery-Funktionen für allgemeine Zwecke werden in der aktuellen Version Ihres Altova-Produkts unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache
Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath, XQuery und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XP1** **XP2** **XP3**
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XSLT1** **XSLT2** **XSLT3**
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	**XQ1** **XQ3**

Funktionen zur automatischen Nummerierung

▼ **generate-auto-number [altova:]**

altova:generate-auto-number(ID *als xs:string*, **StartsWith** *als xs:double*, **Increment** *als xs:double*, **ResetOnChange** *als xs:string*) als `xs:integer` **XP1** **XP2** **XQ1** **XP3** **XQ3**

Generiert jedes Mal, wenn die Funktion aufgerufen wird, eine Zahl. Die erste Zahl, die beim ersten Aufruf der Funktion generiert wird, wird durch das Argument `StartsWith` definiert. Bei jedem erneuten Aufruf der Funktion wird eine neue Zahl generiert. Diese Zahl wird durch den im Argument `Increment` definierten Wert anhand der zuvor generierten Zahl inkrementiert. Auf diese Art erstellt die Funktion `altova:generate-auto-number` einen Zähler, dessen Name durch das Argument `ID` definiert wird und der jedes Mal, wenn die Funktion aufgerufen wird, inkrementiert wird. Wenn sich der Wert des Arguments `ResetOnChange` seit dem vorherigen Funktionsaufruf geändert hat, so wird der Wert der zu generierenden Zahl auf den Wert `StartsWith` zurückgesetzt. Die Automatische Nummerierung kann auch mit der Funktion `altova:reset-auto-number` zurückgesetzt werden.

⊟ *Beispiele*

- **`altova:generate-auto-number`**`("ChapterNumber", 1, 1, "SomeString")` gibt bei jedem Aufruf der Funktion eine einzige Zahl beginnend mit `1` zurück, die bei jedem Aufruf der Funktion um `1` inkrementiert wird. Solange das vierte Argument in jedem anschließenden Aufruf `"SomeString"` bleibt, wird die Inkrementierung fortgesetzt. Wenn sich der Wert des vierten Arguments ändert, wird der Zähler (namens `ChapterNumber`) auf 1 zurückgesetzt. Der Wert von `ChapterNumber` kann auch folgendermaßen durch Aufruf der Funktion `altova:reset-auto-number` zurückgesetzt werden: `altova:reset-auto-number("ChapterNumber")`.

reset-auto-number [altova:]

altova:reset-auto-number(ID *als xs:string***)** XP1 XP2 XQ1 XP3 XQ3

Diese Funktion setzt die Zahl des im `ID`-Argument angegebenen Zählers zur automatischen Nummerierung zurück. Die Zahl wird auf die Zahl zurückgesetzt, die durch das Argument `StartsWith` der Funktion `altova:generate-auto-number`, die den im `ID`-Argument genannten Zähler erstellt hat, definiert ist

Beispiele

- **altova:reset-auto-number**`("ChapterNumber")` setzt die Zahl des Zählers zur automatischen Nummerierung (`ChapterNumber`), der durch die Funktion `altova:generate-auto-number` erstellt wurde, zurück. Die Zahl wird auf den Wert des Arguments `StartsWith` der Funktion `altova:generate-auto-number`, die `ChapterNumber` erstellt hat, zurückgesetzt.

[Nach oben]

Numerische Funktionen

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *als xs:string***)** als **xs:integer** XP3 XQ3

Verwendet ein String-Argument, das das Base-16-Äquivalent einer Ganzzahl im Dezimalsystem (Base-10) ist, und gibt die dezimale Ganzzahl zurück.

Beispiele

- **altova:hex-string-to-integer**`('1')` gibt `1` zurück
- **altova:hex-string-to-integer**`('9')` gibt `9` zurück
- **altova:hex-string-to-integer**`('A')` gibt `10` zurück
- **altova:hex-string-to-integer**`('B')` gibt `11` zurück
- **altova:hex-string-to-integer**`('F')` gibt `15` zurück
- **altova:hex-string-to-integer**`('G')` gibt einen Fehler zurück
- **altova:hex-string-to-integer**`('10')` gibt `16` zurück
- **altova:hex-string-to-integer**`('01')` gibt `1` zurück
- **altova:hex-string-to-integer**`('20')` gibt `32` zurück
- **altova:hex-string-to-integer**`('21')` gibt `33` zurück
- **altova:hex-string-to-integer**`('5A')` gibt `90` zurück
- **altova:hex-string-to-integer**`('USA')` gibt einen Fehler zurück

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer***)** as **xs:string** XP3 XQ3

Verwendet ein Ganzzahlargument und gibt das Base-16-Äquivalent als String zurück.

Beispiele

- **altova:integer-to-hex-string**`(1)` gibt `1` zurück
- **altova:integer-to-hex-string**`(9)` gibt `'9'` zurück
- **altova:integer-to-hex-string**`(10)` gibt `'A'` zurück
- **altova:integer-to-hex-string**`(11)` gibt `'B'` zurück
- **altova:integer-to-hex-string**`(15)` gibt `'F'` zurück
- **altova:integer-to-hex-string**`(16)` gibt `'10'` zurück
- **altova:integer-to-hex-string**`(32)` gibt `'20'` zurück
- **altova:integer-to-hex-string**`(33)` gibt `'21'` zurück

 © 2014 Altova GmbH

- **altova:integer-to-hex-string**(90) gibt '5A' zurück

[Nach oben]

Sequenzfunktionen

attributes [altova:]

altova:attributes(**AttributeName** *als xs:string*) als **attribute()*** **XP3** **XQ3**
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument AttributeName angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der attribute:: Achse durchgeführt wird, beachtet.

Beispiele

- **altova:attributes**("MyAttribute") gibt MyAttribute()* zurück

altova:attributes(**AttributeName** *als xs:string*, **SearchOptions** *als xs:string*) als **attribute()*** **XP3** **XQ3**
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument AttributeName angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der attribute:: Achse durchgeführt wird, beachtet. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
r = wechselt zu einer Suche mittels Regular Expression; bei AttributeName muss es sich in diesem Fall um einen Regular Expression-Suchstring handeln;
i = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
p = inkludiert das Namespace-Präfix in die Suche; AttributeName sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: altova:MyAttribute.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (siehe vorherige Signatur). Unzulässig ist jedoch die Verwendung einer leeren Sequenz.

Beispiele

- **altova:attributes**("MyAttribute", "rip") gibt MyAttribute()* zurück
- **altova:attributes**("MyAttribute", "pri") gibt MyAttribute()* zurück
- **altova:attributes**("MyAttribute", "") gibt MyAttribute()* zurück
- **altova:attributes**("MyAttribute", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:attributes**("MyAttribute",) gibt den Fehler zurück, dass das zweite Argument fehlt.

elements [altova:]

altova:elements(**ElementName** *als xs:string*) als **element()*** **XP3** **XQ3**
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument ElementName angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der child:: Achse durchgeführt wird, beachtet.

Beispiele

- **altova:elements**("MyElement") gibt MyElement()* zurück

altova:elements(ElementName *als xs:string*, **SearchOptions** *als xs:string***)** als **element()*** **XP3 XQ3**

Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument ElementName angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der child:: Achse durchgeführt wird, beachtet. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
r = wechselt zu einer Suche mittels Regular Expression; bei ElementName muss es sich in diesem Fall um einen Regular Expression-Suchstring handeln;
i = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
p = inkludiert das Namespace-Präfix in die Suche; ElementName sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: altova:MyElement.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (siehe vorherige Signatur). Unzulässig ist jedoch die Verwendung einer leeren Sequenz.

Beispiele

- **altova:elements**("MyElement", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElement", "pri") gibt MyElement()* zurück
- **altova:elements**("MyElement", "") gibt MyElement()* zurück
- **altova:elements**("MyElement", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:elements**("MyElement",) gibt den Fehler zurück, dass das zweite Argument fehlt.

find-first [altova:]

altova:find-first((*item()****), (CheckFunction(item()** *als xs:boolean***))** als **item()?** **XP3 XQ3**

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, Condition, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert 1) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in Condition referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von altova:find-first zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 6 zurück

  Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Der erste Input-Wert, der diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first** (in diese Fall **6**) zurückgegeben.

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) gibt xs:integer 4 zurück

Weitere Beispiele

Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt C:\Temp\Customers.xml zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt http://www.altova.com/index.html zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, doc-available, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann true zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.
- Die Funktion doc-available kann für **Condition,** das zweite Argument von altova:find-first verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element item() als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass doc-available nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix #1 gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet doc-available#1: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **doc-available()** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **doc-available()** zu true() ausgewertet und der String wird als Ergebnis der Funktion **altova:find-first** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

find-first-combination [altova:]

altova:find-first-combination((Seq-01 *als item()**), **(Seq-02** *als item()**), **(Condition(Seq-01-Item, Seq-02-Item** *als xs:boolean*)) als **item()*** **XP3** **XQ3**

Diese Funktion verwendet drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit 2 hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in

geordneten Paaren übergeben (je ein Element aus jeder Sequenz bildet ein Paar). Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, wird als Ergebnis von **altova:find-first-combination** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-combination** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **altova:find-first-combination** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt die Sequenz xs:integers (11, 21) zurück
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt die Sequenz xs:integers (11, 22) zurück
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) gibt die Sequenz xs:integers (11, 23) zurück

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *als item()**), **(Seq-02** *als item()**), **(Condition(Seq-01-Item, Seq-02-Item** *als xs:boolean*)) als **item()*** **XP3** **XQ3**

Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, wird als Ergebnis von **altova:find-first-pair** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **altova:find-first-pair** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt die Sequenz xs:integers (11, 21) zurück
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet

sind: `(11, 21) (12, 22) (13, 23)...(20, 30)`. Aus diesem Grund gibt das zweite Beispiel *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe `33` ergibt).

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *als item()**), **(Seq-02** *als item()**), **(Condition(Seq-01-Item, Seq-02-Item** *als xs:boolean*)) als **xs:integer** **XP3** **XQ3**

Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit 2 hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Als Ergebnis von **altova:find-first-pair-pos** wird die Indexposition des ersten geoordneten Paars, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, zurückgegeben. Beachten Sie: Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und kein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair-pos** *Keine Ergebnisse* zurück.

Beispiele

- **altova:find-first-pair**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` gibt `1` zurück
- **altova:find-first-pair**`(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: `(11, 21) (12, 22) (13, 23)...(20, 30)`. Im ersten Beispiel gibt die Funktion **Condition** bei Auswertung des ersten Paars **true()** zurück, daher wird dessen Indexposition in der Sequenz, `1`, zurückgegeben. Das zweite Beispiel gibt *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe `33` ergibt).

find-first-pos [altova:]

altova:find-first-pos((*item()****), (CheckFunction(item()** *als xs:boolean*)) als **xs:integer?** **XP3** **XQ3**

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, `Condition`, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert `1`) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in `Condition` referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition** **true()** ist, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 2 zurück

Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first-pos** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Die Indexposition in der Sequenz des ersten Input-Werts, die diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben (in diesem Fall 2, da **6**, der erste Wert in der Sequenz, der die Bedingung erfüllt, sich in der Sequenz an der Indexposition 2 befindet).

Weitere Beispiele

Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 1 zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 2 zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, doc-available, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann true zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. (Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.)
- Die Funktion doc-available kann für **Condition**, das zweite Argument von altova:find-first-pos verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element item() als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass doc-available nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix #1 gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet doc-available#1: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **doc-available()** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **doc-available()** zu true() ausgewertet und der String wird als Ergebnis der Funktion **altova:find-first** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

▼ substitute-empty [altova:]

altova:substitute-empty(FirstSequence *als item()**, **SecondSequence** *als item()*) als **item()*** **XP3** **XQ3**

Wenn `FirstSequence` leer ist, wird `SecondSequence` zurückgegeben. Wenn `FirstSequence` nicht leer ist, wird `FirstSequence` zurückgegeben.

Beispiele

- **altova:substitute-empty**`( (1,2,3), (4,5,6) )` gibt `(1,2,3)` zurück
- **altova:substitute-empty**`( (), (4,5,6) )` gibt `(4,5,6)` zurück

[Nach oben]

URI-Funktionen

get-temp-folder [altova:]

altova:get-temp-folder() als `xs:string` **XP2** **XQ1** **XP3** **XQ3**

Diese Funktion hat kein Argument. Sie gibt den Pfad zum temporären Ordner des aktuellen Benutzers zurück.

Beispiele

- **altova:get-temp-folder**`()` würde auf einem Windows-Rechner z.B. den folgenden Pfad als `xs:string` zurückgeben: `C:\Users\<UserName>\AppData\Local\Temp\`.

[Nach oben]

9.1.5 Diagrammfunktionen

Mit Hilfe der unten aufgelisteten Diagrammfunktionen können Sie Diagramme als Bilder erstellen, generieren und speichern. Sie werden in der aktuellen Version Ihres Altova-Produkts auf die unten beschriebene Art unterstützt. Beachten Sie jedoch, dass eine oder mehrere dieser Funktionen in zukünftigen Produktversionen eventuell nicht mehr unterstützt werden, bzw. dass sich das Verhalten einzelner Funktionen ändern kann. Um Informationen über die Unterstützung für Altova Erweiterungsfunktionen in der jeweiligen Release zu erhalten, schlagen Sie bitte in der Dokumentation der jeweils aktuellen Release nach.

Die Diagrammfunktionen sind XPath-Funktionen (nicht XSLT-Funktionen) und werden in zwei Gruppen unterteilt:

- Funktionen zum Generieren und Speichern von Diagrammen
- Funktionen zur Erstellung von Diagrammen

Anmerkung: Diagrammfunktionen werden nur in **Altova Server-Produkten** und den **Enterprise-Editionen** von Altova-Produkten unterstützt.

Anmerkung: Unterstützte Bildformate für Diagramme in Server-Editionen sind `jpg`, `png` und `bmp`. Die beste Option ist png, da dabei keine Daten verloren gehen und es sich um ein komprimiertes Format handelt. In Enterprise-Editionen werden die folgenden Formate unterstützt: `jpg`. `png`, `bmp` und `gif`.

Funktionen zum Generieren und Speichern von Diagrammen
Diese Funktionen generieren anhand des (mit Hilfe der Diagrammerstellungsfunktionen) erzeugten Diagrammobjekts entweder ein Bild oder speichern ein Bild in einer Datei

`altova:generate-chart-image` `($chart, $width, $height, $encoding) als atomic`

wobei

- `$chart` ist das Diagrammerweiterungsobjekt, das Sie mit der Funktion `altova:create-chart` erhalten
- `$width` und `$height` muss mit einer Längeneinheit definiert werden
- `$encoding` kann `base64Binary` oder `binarytobase16` sein

Die Funktion gibt das Diagrammbild in der definierten Kodierung zurück.

`altova:generate-chart-image` `($chart, $width, $height, $encoding, $imagetype) als atomic`

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$encoding` den Wert `base64Binary` oder `hexBinary` haben kann
- `$imagetype` eines der folgenden Bildformate sein kann: `png`, `gif`, `bmp`, `jpg`, `jpeg`.Beachten Sie, dass `gif` in Server-Produkten nicht unterstützt wird. *Siehe oben*

 © 2014 Altova GmbH

auf dieser Seite.

Die Funktion gibt das Diagrammbild in der definierten Kodierung und im definierten Bildformat zurück.

---

altova:save-chart-image ($chart, $filename, $width, $height) als empty() ***(nur Windows)***

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$filename` der Pfad und Name der Datei ist, unter dem das Diagrammbild gespeichert werden soll
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen

Die Funktion speichert das Diagrammbild unter dem in `$filename` definierten Dateinamen.

---

altova:save-chart-image ($chart, $filename, $width, $height, $imagetype) als empty() ***(nur Windows)***

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$filename` der Pfad und Name der Datei ist, unter dem das Diagrammbild gespeichert werden soll
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$imagetype` eines der folgenden Bildformate sein kann: `png`, `gif`, `bmp`, `jpg`, `jpeg`. Beachten Sie, dass `gif` in Server-Produkten nicht unterstützt wird. *Siehe oben auf dieser Seite*.

Die Funktion speichert das Diagrammbild im definierten Bildformat unter dem in `$filename` definierten Dateinamen.

Funktionen zur Erstellung von Diagrammen
Die folgenden Funktionen dienen zur Erstellung von Diagrammen.

altova:create-chart($chart-config, $chart-data-series*) als chart extension item

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` ist das chart-data-series Erweiterungsobjekt, das mit der

Funktion `altova:create-chart-data-series` oder mit der Funktion `altova:create-chart-config-from-rows` erzeugt wurde

Die Funktion gibt ein Diagrammerweiterungsobjekt zurück, das anhand der über die Argumente gelieferten Daten erzeugt wird.

`altova:create-chart-config``($type-name, $title) als chart-config Erweiterungsobjekt`

wobei

- `$type-name` den Typ des zu erstellenden Diagramms definiert: `Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge`
- `$title` der Name des Diagramms ist

Die Funktion gibt ein Diagrammkonfigurations-Erweiterungsobjekt zurück, das die Konfigurationsinformationen zum Diagramm enthält.

`altova:create-chart-config-from-xml``($xml-struct) als chart-config Erweiterungsobjekt`

wobei

- `$xml-struct` die XML-Struktur ist, die die Konfigurationsinformationen des Diagramms enthält

Die Funktion gibt ein Diagrammkonfigurations-Erweiterungsobjekt zurück, das die Konfigurationsinformationen zum Diagramm enthält. Diese Informationen werden in einem XML-Datenfragment geliefert.

`altova:create-chart-data-series``($series-name?, $x-values*, $y-values*) als chart-data-series Erweiterungsobjekt`

wobei

- `$series-name` der Name der Datenreihe ist
- `$x-values` die Liste der Werte für die X-Achse liefert
- `$y-values` die Liste der Werte für die Y-Achse liefert

Die Funktion gibt ein Diagrammdatenreihen-Erweiterungsobjekt zurück, das die Daten zur Erstellung des Diagramms, also die Namen der Datenreihen, und die Achsendaten enthält.

 © 2014 Altova GmbH

altova:create-chart-data-row(x, y1, y2, y3, ...) als chart-data-x-*N*y-row Erweiterungsobjekt

wobei

- x der Wert der X-Achsen-Spalte der Diagrammdatenzeile ist
- yN die Werte der Spalten für die Y-Achse sind

Die Funktion gibt ein chart-data-x-Ny-row Erweiterungsobjekt zurück, das die Daten für die X-Achsen-Spalte und die Y-Achsen-Spalten einer einzigen Datenreihe enthält.

---

altova:create-chart-data-series-from-rows($series-names als xs:string*, $row*) als chart-data-series Erweiterungsobjekt

wobei

- $series-name der Name der zu erstellenden Datenreihen ist
- $row das chart-data-x-Ny-row Erweiterungsobjekt ist, das als Datenreihe erstellt werden soll

Die Funktion gibt ein chart-data-series Erweiterungsobjekt zurück, das die Daten für die X- und die Y-Achse der Datenreihe enthält.

---

altova:create-chart-layer($chart-config, $chart-data-series*) als chart-layer Erweiterungsobjekt

wobei

- $chart-config das chart-config-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-config oder über die Funktion altova:create-chart-config-from-xml abgerufen wird
- $chart-data-series das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-data-series oder der Funktion altova:create-chart-data-series-from-rows abgerufen wird

Die Funktion gibt ein chart-layer Erweiterungsobjekt zurück, das chart-layer-Daten enthält.

---

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*)

wobei

- $chart-config das chart-config Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-config oder über die Funktion altova:create-chart-config-from-xml abgerufen wird
- $chart-data-series das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-data-series oder der Funktion altova:create-

`chart-data-series-from-rows` abgerufen wird
- `$chart-layer` das chart-layer-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-layer` abgerufen wird

Die Funktion gibt ein multi-layer-chart-Objekt zurück.

`altova:create-multi-layer-chart``($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)`

wobei

- `$chart-config` das chart-config Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` abgerufen wird
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird
- `$chart-layer` das chart-layer-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-layer` abgerufen wird

Die Funktion gibt ein multi-layer-chart-Objekt zurück.

XML-Struktur von Diagrammdaten

Unten sehen Sie die XML-Struktur von Diagrammdaten, wie sie für Altova-Erweiterungsfunktionen für Diagramme angezeigt werden könnte. Diese Funktionen beeinflussen das Aussehen der einzelnen Diagramme. Nicht alle Elemente werden für alle Diagrammarten verwendet, so wird z.B. das Element <Pie> bei Balkendiagrammen ignoriert.

Anmerkung: Diagrammfunktionen werden nur in **Enterprise und Server-Editionen** von Altova Produkten unterstützt.

```
<chart-config>
   <General
      SettingsVersion="1"  must be provided
      ChartKind="BarChart"  Pie, Pie3d, BarChart, StackedBarChart, BarChart3d,
BarChart3dGrouped, LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge,
BarGauge, CandleStick
      BKColor="#ffffff"  Color
      BKColorGradientEnd="#ffffff"  Color. In case of a gradient, BKColor and
BKColorGradientEnd define the gradient's colors
      BKMode="#ffffff"  Solid, HorzGradient, VertGradient
      BKFile="Path+Filename"  String. If file exists, its content is drawn over the
background.
      BKFileMode="Stretch"  Stretch, ZoomToFit, Center, Tile
      ShowBorder="1"  Bool
      PlotBorderColor="#000000"  Color
      PlotBKColor="#ffffff"  Color
      Title=""  String
```

```
    ShowLegend="1"   Bool
    OutsideMargin="3.%"   PercentOrPixel
    TitleToPlotMargin="3.%"   PercentOrPixel
    LegendToPlotMargin="3.%"   PercentOrPixel
    Orientation="vert"   Enumeration: possible values are: vert, horz
    >
    <TitleFont
       Color="#000000"   Color
       Name="Tahoma"   String
       Bold="1"   Bool
       Italic="0"   Bool
       Underline="0"   Bool
       MinFontHeight="10.pt"   FontSize (only pt values)
       Size="8.%"   FontSize />
    <LegendFont
       Color="#000000"
       Name="Tahoma"
       Bold="0"
       Italic="0"
       Underline="0"
       MinFontHeight="10.pt"
       Size="3.5%" />
    <AxisLabelFont
       Color="#000000"
       Name="Tahoma"
       Bold="1"
       Italic="0"
       Underline="0"
       MinFontHeight="10.pt"
       Size="5.%" />
</General>

<Line
    ConnectionShapeSize="1.%"   PercentOrPixel
    DrawFilledConnectionShapes="1"   Bool
    DrawOutlineConnectionShapes="0"   Bool
    DrawSlashConnectionShapes="0"   Bool
    DrawBackslashConnectionShapes="0"   Bool
/>

<Bar
    ShowShadow="1"   Bool
    ShadowColor="#a0a0a0"   Color
    OutlineColor="#000000"   Color
    ShowOutline="1"   Bool
/>

<Area
    Transparency="0"   UINT ( 0-255 ) 255 is fully transparent, 0 is opaque
    OutlineColor="#000000"   Color
    ShowOutline="1"   Bool
/>

<CandleStick
```

```
        FillHighClose="0"   Bool. If 0, the body is left empty. If 1, FillColorHighClose is used
for the candle body
        FillColorHighClose="#ffffff"   Color. For the candle body when close > open
        FillHighOpenWithSeriesColor="1"   Bool. If true, the series color is used to fill the
candlebody when open > close
        FillColorHighOpen="#000000"   Color. For the candle body when open > close and
FillHighOpenWithSeriesColor is false
    />

    <Colors   User-defined color scheme: By default this element is empty except for the style
and has no Color attributes
        UseSubsequentColors ="1"  Boolean. If 0, then color in overlay is used. If 1, then
subsequent colors from previous chart layer is used
        Style="User"   Possible values are: "Default", "Grayscale", "Colorful", "Pastel", "User"
        Colors="#52aca0"   Color: only added for user defined color set
        Colors1="#d3c15d"   Color: only added for user defined color set
        Colors2="#8971d8"   Color: only added for user defined color set
        ...
        ColorsN=""   Up to ten colors are allowed in a set: from Colors to Colors9
    </Colors>

    <Pie
        ShowLabels="1"   Bool
        OutlineColor="#404040"   Color
        ShowOutline="1"   Bool
        StartAngle="0."   Double
        Clockwise="1"   Bool
        Draw2dHighlights="1"   Bool
        Transparency="0"   Int (0 to 255: 0 is opaque, 255 is fully transparent)
        DropShadowColor="#c0c0c0"   Color
        DropShadowSize="5.%"   PercentOrPixel
        PieHeight="10.%"   PercentOrPixel. Pixel values might be different in the result
because of 3d tilting
        Tilt="40.0"   Double (10 to 90: The 3d tilt in degrees of a 3d pie)
        ShowDropShadow="1"   Bool
        ChartToLabelMargin="10.%"   PercentOrPixel
        AddValueToLabel="0"   Bool
        AddPercentToLabel="0"   Bool
        AddPercentToLabels_DecimalDigits="0"  UINT ( 0 – 2 )
        >
        <LabelFont
            Color="#000000"
            Name="Arial"
            Bold="0"
            Italic="0"
            Underline="0"
            MinFontHeight="10.pt"
            Size="4.%"/>
    </Pie>

    <XY>
        <XAxis   Axis
            AutoRange="1"   Bool
            AutoRangeIncludesZero="1"   Bool
```

```
        RangeFrom="0."   Double: manual range
        RangeTill="1."   Double : manual range
        LabelToAxisMargin="3.%"   PercentOrPixel
        AxisLabel=""   String
        AxisColor="#000000"   Color
        AxisGridColor="#e6e6e6"   Color
        ShowGrid="1"   Bool
        UseAutoTick="1"   Bool
        ManualTickInterval="1."   Double
        AxisToChartMargin="0.px"   PercentOrPixel
        TickSize="3.px"   PercentOrPixel
        ShowTicks="1"   Bool
        ShowValues="1"   Bool
        AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
        AxisPositionAtValue = "0"   Double
        >
        <ValueFont
            Color="#000000"
            Name="Tahoma"
            Bold="0"
            Italic="0"
            Underline="0"
            MinFontHeight="10.pt"
            Size="3.%"/>
    </XAxis>
    <YAxis   Axis (same as for XAxis)
        AutoRange="1"
        AutoRangeIncludesZero="1"
        RangeFrom="0."
        RangeTill="1."
        LabelToAxisMargin="3.%"
        AxisLabel=""
        AxisColor="#000000"
        AxisGridColor="#e6e6e6"
        ShowGrid="1"
        UseAutoTick="1"
        ManualTickInterval="1."
        AxisToChartMargin="0.px"
        TickSize="3.px"
        ShowTicks="1"   Bool
        ShowValues="1"   Bool
        AxisPosition="LeftOrBottom"   Enums: "LeftOrBottom", "RightOrTop", "AtValue"
        AxisPositionAtValue = "0"   Double
        >
        <ValueFont
            Color="#000000"
            Name="Tahoma"
            Bold="0"
            Italic="0"
            Underline="0"
            MinFontHeight="10.pt"
            Size="3.%"/>
    </YAxis>
</XY>
```

<XY3d

AxisAutoSize="1" *Bool: If false, XSize and YSize define the aspect ration of x and y axis. If true, aspect ratio is equal to chart window*

XSize="100.%" *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

YSize="100.%" *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

SeriesMargin="30.%" *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

Tilt="20." *Double. -90 to +90 degrees*

Rot="20." *Double. -359 to +359 degrees*

FoV="50."> *Double. Field of view: 1-120 degree*

```
>
<ZAxis
    AutoRange="1"
    AutoRangeIncludesZero="1"
    RangeFrom="0."
    RangeTill="1."
    LabelToAxisMargin="3.%"
    AxisLabel=""
    AxisColor="#000000"
    AxisGridColor="#e6e6e6"
    ShowGrid="1"
    UseAutoTick="1"
    ManualTickInterval="1."
    AxisToChartMargin="0.px"
    TickSize="3.px" >
    <ValueFont
        Color="#000000"
        Name="Tahoma"
        Bold="0"
        Italic="0"
        Underline="0"
        MinFontHeight="10.pt"
        Size="3.%"/>
</ZAxis>
```

</XY3d>

<Gauge

MinVal="0." *Double*

MaxVal="100." *Double*

MinAngle="225" *UINT: -359-359*

SweepAngle="270" *UINT: 1-359*

BorderToTick="1.%" *PercentOrPixel*

MajorTickWidth="3.px" *PercentOrPixel*

MajorTickLength="4.%" *PercentOrPixel*

MinorTickWidth="1.px" *PercentOrPixel*

MinorTickLength="3.%" *PercentOrPixel*

BorderColor="#a0a0a0" *Color*

FillColor="#303535" *Color*

MajorTickColor="#a0c0b0" *Color*

MinorTickColor="#a0c0b0" *Color*

BorderWidth="2.%" *PercentOrPixel*

NeedleBaseWidth="1.5%" *PercentOrPixel*

NeedleBaseRadius="5.%" *PercentOrPixel*

 © 2014 Altova GmbH

```
        NeedleColor="#f00000"   Color
        NeedleBaseColor="#141414"   Color
        TickToTickValueMargin="5.%"   PercentOrPixel
        MajorTickStep="10."   Double
        MinorTickStep="5."   Double
        RoundGaugeBorderToColorRange="0.%"   PercentOrPixel
        RoundGaugeColorRangeWidth ="6.%"   PercentOrPixel
        BarGaugeRadius="5.%"   PercentOrPixel
        BarGaugeMaxHeight="20.%"   PercentOrPixel
        RoundGaugeNeedleLength="45.%"   PercentOrPixel
        BarGaugeNeedleLength="3.%"   PercentOrPixel
        >
        <TicksFont
            Color="#a0c0b0"
            Name="Tahoma"
            Bold="0"
            Italic="0"
            Underline="0"
            MinFontHeight="10.pt"
            Size="4.%"
        />
        <ColorRanges>   User-defined color ranges. By default empty with no child element
entries
            <Entry
                From="50. "   Double
                FillWithColor="1"   Bool
                Color="#00ff00"   Color
            />
            <Entry
                From="50.0"
                FillWithColor="1"
                Color="#ff0000"
            />
            ...
        </ColorRanges>
    </Gauge>
</chart-config>
```

Beispiel: Diagrammfunktionen

Anhand des XSLT-Beispieldokuments weiter unten sehen Sie, wie Altova-Erweiterungsfunktionen für Diagramme eingesetzt werden können. Weiter unten sehen Sie ein XML-Dokument und eine Abbildung des Ausgabebilds, das generiert wird, wenn ein XML-Dokument mit dem XSLT 2.0- oder XSLT 3.0-Prozessor anhand des XSLT-Dokuments verarbeitet wird.

Anmerkung: Diagrammfunktionen werden nur in **Enterprise und Server-Editionen** von Altova Produkten unterstützt.

Anmerkung: Weitere Informationen zur Erstellung von Diagrammdatentabellen finden Sie in der Dokumentation zu den Altova-Produkten XMLSpy und StyleVision.

XSLT-Dokument

In diesem (*unten aufgelisteten*) XSLT-Dokument wird mit Hilfe der Altova Diagramm-Erweiterungsfunktionen ein Kreisdiagramm generiert. Das XSLT-Dokument kann zur Verarbeitung des weiter unten angeführten XML-Dokuments verwendet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:altovaext="http://www.altova.com/xslt-extensions"
    exclude-result-prefixes="#all">
    <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>
                    <xsl:text>HTML Page with Embedded Chart</xsl:text>
                </title>
            </head>
            <body>
                <xsl:for-each select="/Data/Region[1]">
                    <xsl:variable name="extChartConfig" as="item()*">
                        <xsl:variable name="ext-chart-settings" as="item()*">
                            <chart-config>
                                <General
                                    SettingsVersion="1"
                                    ChartKind="Pie3d"
                                    BKColor="#ffffff"
                                    ShowBorder="1"
                                    PlotBorderColor="#000000"
                                    PlotBKColor="#ffffff"
                                    Title="{@id}"
                                    ShowLegend="1"
                                    OutsideMargin="3.2%"
                                    TitleToPlotMargin="3.%"
                                    LegendToPlotMargin="6.%"
                                    >
                                    <TitleFont
                                        Color="#023d7d"
                                        Name="Tahoma"
                                        Bold="1"
                                        Italic="0"
                                        Underline="0"
                                        MinFontHeight="10.pt"
                                        Size="8.%" />
                                </General>
                            </chart-config>
                        </xsl:variable>
                        <xsl:sequence select="altovaext:create-chart-config-from-
xml( $ext-chart-settings )"/>
                    </xsl:variable>
                    <xsl:variable name="chartDataSeries" as="item()*">
                        <xsl:variable name="chartDataRows" as="item()*">
```

 © 2014 Altova GmbH

```
                        <xsl:for-each select="(Year)">
                            <xsl:sequence select="altovaext:create-chart-data-
row( (@id), ( .) )"/>
                        </xsl:for-each>
                    </xsl:variable>
                    <xsl:variable name="chartDataSeriesNames" as="xs:string*"
select=" ( (&quot;Series 1&quot;), &apos;&apos; )[1]"/>
                    <xsl:sequence
                        select="altovaext:create-chart-data-series-from-
rows( $chartDataSeriesNames, $chartDataRows)"/>
                </xsl:variable>
                <xsl:variable name="ChartObj" select="altovaext:create-
chart( $extChartConfig, ( $chartDataSeries), false() )"/>
                <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
                <img src="{$sChartFileName, altovaext:save-chart-
image( $ChartObj, $sChartFileName, 400, 400 ) }"/>
            </xsl:for-each>
        </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XML-Dokument

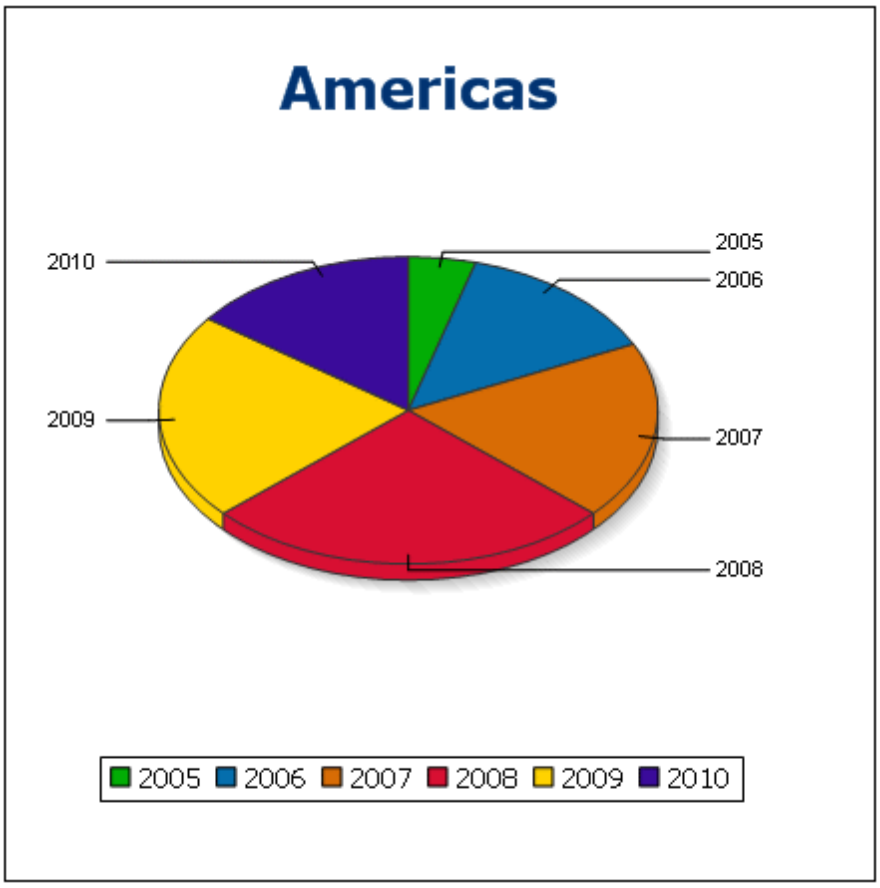
Dieses XML-Dokument kann mit dem oben stehenden XSLT-Dokument verarbeitet werden. Anhand der Daten im XML-Dokument wird das in der unten stehenden Abbildung gezeigte Kreisdiagramm generiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <ChartType>Pie Chart 2D</ChartType>
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
      <Year id="2006">25000</Year>
      <Year id="2007">70000</Year>
      <Year id="2008">110000</Year>
```

```
        <Year id="2009">125000</Year>
        <Year id="2010">150000</Year>
    </Region>
</Data>
```

Ausgabebild

Das unten gezeigt Kreisdiagramm wird generiert, wenn das oben aufgelistete XML-Dokument mit Hilfe des XSLT-Dokuments verarbeitet wird.



 © 2014 Altova GmbH

9.1.6 Barcode-Funktionen

Der XSLT-Prozessor verwendet zur Erstellung von Barcodes Java-Bibliotheken von Drittanbietern. Im Folgenden finden Sie die verwendeten Klassen und öffentlichen Methoden. Die Klassen befinden sich im Paket `AltovaBarcodeExtension.jar`, das im Ordner *<ProgramFilesFolder>*`\Altova\Common2015\jar`.
gespeichert ist.

Die verwendeten Java-Bibliotheken befinden sich in Unterordnern des Ordners *<ProgramFilesFolder>*`\Altova\Common2015\jar`.

- `barcode4j\barcode4j.jar` (Website: http://barcode4j.sourceforge.net/)
- `zxing\core.jar` (Website: http://code.google.com/p/zxing/)

Die Lizenzdateien befinden sich ebenfalls in den entsprechenden Ordnern.

Das Paket `com.altova.extensions.barcode`
Das Paket `com.altova.extensions.barcode` wird zum Generieren der meisten der Barcode-Typen verwendet.

Die folgenden Klassen werden verwendet:

```
public class BarcodeWrapper
    static BarcodeWrapper newInstance( String name, String msg, int dpi, int
orientation, BarcodePropertyWrapper[] arrProperties )
    double getHeightPlusQuiet()
    double getWidthPlusQuiet()
    org.w3c.dom.Document generateBarcodeSVG()
    byte[] generateBarcodePNG()
    String generateBarcodePngAsHexString()
```

`public class` **`BarcodePropertyWrapper`** *Dient zum Speichern der Barcode-Eigenschaften, die zu einem späteren Zeitpunkt dynamisch definiert werden*

```
    BarcodePropertyWrapper( String methodName, String propertyValue )
    BarcodePropertyWrapper( String methodName, Integer propertyValue )

    BarcodePropertyWrapper( String methodName, Double propertyValue )
    BarcodePropertyWrapper( String methodName, Boolean propertyValue )
    BarcodePropertyWrapper( String methodName, Character propertyValue )
    String getMethodName()
    Object getPropertyValue()
```

`public class` **`AltovaBarcodeClassResolver`** *Registriert die Klasse class* `com.altova.extensions.barcode.proxy.zxing.QRCodeBean` *zusätzlich zu den vom* `org.krysalis.barcode4j.DefaultBarcodeClassResolver` *registrierten Klassen für die* `qrcode` *Bean.*

Das Paket `com.altova.extensions.barcode.proxy.zxing`

Das Paket `com.altova.extensions.barcode.proxy.zxing` wird zum Generieren des QRCode Barcodetyps verwendet.

Die folgenden Klassen werden verwendet:

Klasse **`QRCodeBean`**

- *Erweitert* `org.krysalis.barcode4j.impl.AbstractBarcodeBean`
- *Erstellt ein* `AbstractBarcodeBean` Interface für `com.google.zxing.qrcode.encoder`

```
void generateBarcode(CanvasProvider canvasImp, String msg)
void setQRErrorCorrectionLevel(QRCodeErrorCorrectionLevel level)
BarcodeDimension calcDimensions(String msg)
double getVerticalQuietZone()
double getBarWidth()
```

Klasse **`QRCodeErrorCorrectionLevel`** *Fehlerkorrekturebene für den* `QRCode`

```
static QRCodeErrorCorrectionLevel byName(String name)
"L" = ~7% correction
"M" = ~15% correction
"H" = ~25% correction
"Q" = ~30% correction
```

XSLT-Beispiel

Im Folgenden sehen Sie ein XSLT-Beispiel für die Verwendung von Barcode-Funktionen in einem XSLT Stylesheet.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:altova="http://www.altova.com"
   xmlns:altovaext="http://www.altova.com/xslt-extensions"
   xmlns:altovaext-barcode="java:com.altova.extensions.barcode.BarcodeWrapper"
   xmlns:altovaext-barcode-
property="java:com.altova.extensions.barcode.BarcodePropertyWrapper">
   <xsl:output method="html" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <html>
         <head><title/></head>
         <body>
            <img alt="barcode" src="{altovaext:get-temp-folder()}barcode.png"/>
         </body>
      </html>
      <xsl:result-document
         href="{altovaext:get-temp-folder()}barcode.png"
         method="text" encoding="base64tobinary" >
         <xsl:variable name="barcodeObject"
            select="altovaext-
barcode:newInstance(&apos;Code39&apos;,string(&apos;some value&apos;),
            96,0, (altovaext-barcode-property:new( &apos;setModuleWidth&apos;,
25.4 div 96 * 2 ) ) )"/>
```

 © 2014 Altova GmbH

```
            <xsl:value-of select="xs:base64Binary(xs:hexBinary(string(altovaext-
barcode:generateBarcodePngAsHexString($barcodeObject)) ))"/>
         </xsl:result-document>
      </xsl:template>
</xsl:stylesheet>
```

9.2 Diverse Erweiterungsfunktionen

Es gibt in Programmiersprachen wie Java und C# eine Reihe von fertigen Funktionen, die nicht als XQuery / XPath 2.0- oder XSLT-Funktionen zur Verfügung stehen. Ein gutes Beispiel dafür sind die mathematischen in Java verfügbaren Funktionen wie z.B. `sin()` und `cos()`. Stünden diese Funktionen für die Erstellung von XSLT Stylesheets und XQuery-Abfragen zur Verfügung, würde sich der Einsatzbereich von Stylesheets und Abfragen erweitern und die Erstellung von Stylesheets wäre viel einfacher. Der in einer Reihe von Altova-Produkten verwendete XSLT- und XQuery-Prozessor von Altova unterstützt die Verwendung von Erweiterungsfunktionen in Java und .NET sowie MSXSL Skripts für XSLT. MSXSL-Skripts für XSLT und die Altova-Erweiterungsfunktionen. In diesem Abschnitt wird beschrieben, wie Sie Erweiterungsfunktionen und MSXSL-Skripts in Ihren XSLT Stylesheets und XQuery-Dokumenten verwenden können. Diese Beschreibungen finden Sie in den folgenden Abschnitten:

- Java-Erweiterungsfunktionen
- .NET-Erweiterungsfunktionen
- MSXSL-Skripts für XSLT

Hauptsächlich werden dabei die folgenden beiden Punkte behandelt: (i) Wie Funktionen in den entsprechenden Bibliotheken aufgerufen werden; und (ii) welche Regeln beim Konvertieren von Argumenten in einem Funktionsaufruf in das erforderliche Format der Funktion befolgt werden und welche Regeln bei der Rückwärtskonvertierung (Funktionsresultat in XSLT/XQuery Datenobjekt) befolgt werden.

Voraussetzungen
Damit die Erweiterungsfunktionen unterstützt werden, muss auf dem Rechner, auf dem die XSLT-Transformation oder die XQuery-Ausführung stattfindet, eine Java Runtime-Umgebung (zum Aufrufen der Java-Funktionen) und ein .NET Framework 2.0 (Mindestvoraussetzung für Zugriff auf .NET-Funktionen) installiert sein oder es muss Zugriff auf eine solche bestehen.

© 2014 Altova GmbH

9.2.1 Java-Erweiterungsfunktionen

Eine Java-Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Java-Konstruktor oder eine Java-Methode (statisch oder Instanz) aufzurufen.

Ein Feld in einer Java-Klasse wird als Methode ohne Argument betrachtet. Bei einem Feld kann es sich um ein statisches Feld oder eine Instanz handeln. Wie man Felder aufruft, wird in den entsprechenden Unterabschnitten zu statischen Feldern und Instanzen beschrieben.

Dieser Abschnitt enthält die folgenden Unterabschnitte:

- Java: Konstruktoren
- Java: Statische Methoden und statische Felder
- Java: Instanzmethoden und Instanzfelder
- Datentypen: XPath/XQuery in Java
- Datentypen: Java in XPath/XQuery

Form der Erweiterungsfunktion
Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgenden Form haben `präfix:fname()`.

- Der Teil `präfix:` kennzeichnet die Erweiterungsfunktion als Java-Funktion, indem er die Erweiterungsfunktion mit einer in-scope Namespace-Deklaration verknüpft, deren URI mit `java:` beginnen muss (*Beispiele siehe unten*). Die Namespace-Deklaration sollte eine Java-Klasse bezeichnen, z.B: `xmlns:myns="java:java.lang.Math"`. Sie könnte aber auch einfach lauten: `xmlns:myns="java"` (ohne Doppelpunkt), wobei die Identifizierung der Java-Klasse dem `fname()` Teil der Erweiterungsfunktion überlassen bleibt.
- Der Teil `fname()` identifiziert die aufgerufene Java-Methode und liefert die Argumente für die Methode (*Beispiele siehe unten*). Wenn die durch das `prefix:` Teil identifizierte Namespace URI jedoch keine Java-Klasse bezeichnet (*siehe vorheriger Punkt),* dann sollte die Java-Klasse im `fname()` Teil vor der Klasse identifiziert werden und von der Klasse durch einen Punkt getrennt sein (*siehe zweites XSLT-Beispiel unten*).

Anmerkung: Die aufgerufene Klasse muss sich unter dem Klassenpfad des Rechners befinden.

XSLT-Beispiel
Hier sehen Sie zwei Beispiele dafür, wie eine statische Methode aufgerufen werden kann. Im ersten Beispiel ist der Klassenname (`java.lang.Math`) in der Namespace URI enthalten und darf daher nicht im `fname()` Teil enthalten sein. Im zweiten Beispiel liefert der `prefix:` Teil das Präfix `java:`, während der `fname()` Teil die Klasse sowie die Methode identifiziert.

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

Die in der Erweiterungsfunktion (im Beispiel oben `cos()`) angegebene Methode muss mit dem Namen einer öffentlichen statischen Methode in der angegebenen Java-Klasse (im Beispiel oben `java.lang.Math`) übereinstimmen.

XQuery-Beispiel
Hier sehen Sie ein XQuery-Beispiel, das dem XSLT-Beispiel oben ähnlich ist:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Benutzerdefinierte Java-Klassen
Wenn Sie Ihre eigenen Java-Klassen erstellt haben, werden die Methoden in diesen Klassen unterschiedlich aufgerufen, je nachdem: (i) ob die Klassen über eine JAR-Datei oder eine Klassendatei aufgerufen werden, und (ii) ob sich diese Dateien (JAR oder Klasse) im aktuellen Verzeichnis befinden (im selben Verzeichnis wie das XSLT- oder XQuery-Dokument) oder nicht. Wie Sie diese Dateien finden, wird in den Abschnitten Benutzerdefinierte Klassendateien und Benutzerdefinierte Jar-Dateien beschrieben. Pfade zu Klassendateien, die sich nicht im aktuellen Verzeichnis befinden, und Pfade zu allen JAR-Dateien müssen jedoch angegeben werden.

Benutzerdefinierte Klassendateien

Wenn der Zugriff über eine Klassendatei erfolgt, gibt es zwei Möglichkeiten:

- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie das Java-Paket. (*Siehe Beispiel unten*.)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie die Klassendatei. (*Siehe Beispiel unten*.)
- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten*.)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten*.)

Gesetzt der Fall, die Klassendatei befindet sich nicht in einem Paket, sondern im selben Ordner wie das XSLT- oder XQuery-Dokument, so muss der Dateipfad nicht angegeben werden, da alle Klassen im Ordner gefunden werden. Die Syntax zum Identifizieren einer Klasse lautet:

```
java:classname
```

wobei

`java`: angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird; (Java-Klassen im aktuellen Verzeichnis werden standardmäßig geladen)
`classname` der Name der Klasse der erforderlichen Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird.

 © 2014 Altova GmbH

Klassendatei in einem Paket, XSLT/XQuery-Datei befindet sich im selben Ordner wie das Java-Paket

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject`.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei befindet sich nicht in einem Paket, die XSLT/XQuery-Datei befindet sich im selben Ordner wie die Klassendatei

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Die Klassendatei `Car` class befindet sich im folgenden Ordner: `JavaProject/com/altova/extfunc`. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject/com/altova/extfunc`.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei befindet sich in einem Paket, die XSLT/XQuery-Datei befindet sich in

einem beliebigen Ordner
Im Beispiel unten wird die Methode `getCarColor()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. In diesem Fall muss der Pfad des Pakets mit der URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=uri-of-classfile]
```

wobei

`java`: angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/
JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei befindet sich nicht in einem Paket, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner
Im Beispiel unten wird die Methode `getCarColor()` der Klasse `Car` class des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. Der Pfad der Klassendatei wird in der Namespace-URI als Abfragestring definiert. Die Syntax lautet:

```
java:classname[?path=uri-of-classfile]
```

wobei

`java`: angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem

 © 2014 Altova GmbH

Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/
extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

Benutzerdefinierte Jar-Dateien

JAR-Dateien

Wenn der Zugriff über eine JAR-Datei erfolgt, muss die URI der JAR-Datei mit Hilfe der folgenden Syntax definiert werden:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

Die Methode wird anschließend durch Verwendung des Präfix der Namespace URI aufgerufen, der die Klasse bezeichnet: `classNS:method()`

wobei im obigen Beispiel:

`java:` angibt, dass eine Java-Funktion aufgerufen wird
`classname` der Name der Klasse der benutzerdefinierten Klasse ist
`?` das Trennzeichen zwischen dem Klassennamen und dem Pfad ist
`path=jar:` angibt, dass es sich um einen Pfad zu einer JAR-Datei handelt
`uri-of-jarfile` die URI der jar-Datei angibt
`!/` das Trennzeichen am Ende des Pfades ist
`classNS:method()` der Aufruf der Methode ist

Alternativ dazu kann der Klassenname mit dem Methodenaufruf angegeben werden. Hier sehen Sie zwei Beispiele für die Syntax:

```
xmlns:ns1="java:docx.layout.pages?path=jar:file:///c:/projects/docs/
docx.jar!/"
ns1:main()

xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
ns2:docx.layout.pages.main()
```

Hier sehen Sie ein komplettes XSLT-Beispiel, in dem eine JAR-Datei verwendet wird, um eine Java-Erweiterungsfunktion aufzurufen.:

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:fn="http://www.w3.org/2005/xpath-functions"
      xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
      <xsl:variable name="myCar" select="car:Car1.new('red')" />
      <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

Java: Konstruktoren

Eine Erweiterungsfunktion kann dazu verwendet werden, um einen Java-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen.

Wenn das Ergebnis eines Java-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann, dann gibt die Java-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einem XPath/XQuery-Datentyp handelt. Wenn das Ergebnis eines Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, dann erstellt der Konstruktor ein wrapped Java-Objekt mit einem Typ, der den Namen der Klasse hat, die dieses Java-Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `java.util.Date` aufgerufen wird (`java.util.Date.new()`), so wird ein Objekt vom Typ `java.util.Date` zurückgegeben. Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format des XPath-Datentyps überein und der Wert müsste daher in das lexikalische Format des erforderlichen XPath-Datentyps und anschließend in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes Java-Objekt kann für zwei Zwecke verwendet werden:

- Es kann einer Variable zugewiesen werden:
  ```
  <xsl:variable name="currentdate" select="date:new()"
  xmlns:date="java:java.util.Date" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe Instanzmethode und Instanzfelder*):
  ```
  <xsl:value-of select="date:toString(date:new())"
  xmlns:date="java:java.util.Date" />
  ```

 © 2014 Altova GmbH

Java: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Java-Namen und durch Angabe der Argumente für die Methode aufgerufen. Statische Felder (Methoden, die keine Argumente haben), wie z.B. die Konstantenwertfelder `E` und `PI` werden ohne Angabe eines Arguments aufgerufen.

XSLT-Beispiele
Hier sehen Sie einige Beispiele dafür, wie statische Methoden und Felder aufgerufen werden können:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

Beachten Sie, dass die Erweiterungsfunktionen die Form `prefix:fname()` haben. Das Präfix ist in allen drei Fällen `jMath:`. Es ist mit der Namespace URI `java:java.lang.Math` verknüpft. (Die Namespace URI muss mit `java:`. beginnen. In den obigen Beispielen wurde es um den Klassennamen erweitert (`java.lang.Math`).) Der Teil `fname()` der Erweiterungsfunktionen muss mit dem Namen der öffentlichen Klasse (z.B. `java.lang.Math`) gefolgt vom Namen einer öffentlichen statischen Methode mit ihrem/ihren Argument(en) (wie z.B. `(3.14)`) oder einem öffentlichen statischen Feld (z.B. `PI()`) übereinstimmen.

In den obigen Beispielen wurde der Klassenname in die Namespace URI inkludiert. Wäre sie nicht in der Namespace URI enthalten, müsste sie in den `fname()` Teil der Erweiterungsfunktion inkludiert werden. Z.B:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery-Beispiel
Ein ähnliches Beispiel in XQuery wäre:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Java: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument eines Methodenaufrufs ein Java-Objekt an die Methode übergeben. Ein solches Java-Objekt würde normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. eines Konstruktoraufrufs) oder eines Stylesheet-Parameters/einer Stylesheet-Variablen erstellt. Ein XSLT-Beispiel dafür wäre:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="
{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird der Wert des Node `enrollment/@type` folgendermaßen erstellt:

1. Es wird ein Objekt mit einem Konstruktor für die Klasse `java.util.Date` (mit dem Konstruktor `date:new()`) erstellt.
2. Dieses Java-Objekt wird als das Argument der Methode `jlang.Object.getClass` übergeben.
3. Das mit der Methode `getClass` abgerufene Objekt wird als das Argument an die Methode `jlang.Object.toString` übergeben.

Das Ergebnis (der Wert von `@type`) ist ein String, der den Wert `java.util.Date` hat.

Ein Instanzfeld unterscheidet sich theoretisch insofern von einer Instanzmethode, als es sich nicht um ein Java-Objekt per se handelt, das als Argument an das Instanzfeld übergeben wird. Stattdessen wird ein Parameter oder eine Variable als Argument übergeben. Der Parameter/die Variable kann allerdings selbst den Wert enthalten, der von einem Java-Objekt zurückgegeben wird. So erhält z.B. der Parameter `CurrentDate` den Wert, der von einem Konstruktor für die Klasse `java.util.Date` zurückgegeben wird. Dieser Wert wird anschließend als Argument an die Instanzmethode `date:toString` übergeben, um den Wert von `/enrollment/@date` bereitzustellen.

Datentypen: XPath/XQuery in Java

Wenn von einem XPath/XQuery-Ausdruck aus eine Java-Funktion aufgerufen wird, spielt der Datentyp der Argumente der Funktion eine wichtige Rolle, welche von mehreren Java-Klassen desselben Namens aufgerufen wird.

In Java gelten die folgenden Regeln:

- Wenn es mehr als eine Java-Methode mit demselben Namen gibt, jede aber eine andere Anzahl von Argumenten als die andere(n) hat, so wird die Java-Methode ausgewählt, die der Anzahl der Argumente im Funktionsaufruf am ehesten entspricht.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden Java-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen Java-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener Java-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene Java-Methode z.B. `fx(decimal)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den Java-Datentyp `decimal` konvertiert.

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in Java-Datentypen.

xs:string	java.lang.String
xs:boolean	boolean (primitive), java.lang.Boolean
xs:integer	int, long, short, byte, float, double und die Wrapper-Klassen davon wie z.B. java.lang.Integer
xs:float	float (primitive), java.lang.Float, double (primitive)
xs:double	double (primitive), java.lang.Double
xs:decimal	float (primitive), java.lang.Float, double(primitive), java.lang.Double

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die Java-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige Java-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein xs:untypedAtomic Wert 10 und ist für die Methode mymethod(float) bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: mymethod(double).
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (xs:untypedAtomic) sowohl in float als auch double korrekt konvertiert werden könnte, kann es geschehen, dass xs:untypedAtomic in double anstelle von float konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. xs:date), werden nicht konvertiert und generieren einen Fehler. Beachten Sie jedoch, dass es in einigen Fällen unter Umständen möglich ist, den benötigten Java-Typ mittels eines Java-Konstruktors zu erstellen.

Datentypen: Java in XPath/XQuery

Wenn eine Java-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So werden z.B. die Java-Datentypen javaa.lang.Boolean und boolean in xsd:boolean konvertiert.

Von Funktionen zurückgegebene eindimensionale Arrays werden zu einer Sequenz erweitert. Mehrdimensionale Arrays werden nicht konvertiert und sollten daher in einen Wrapper gesetzt werden.

Wenn ein wrapped Java-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine Java-Methode (e.g toString) verwenden, um das Java-Objekt in einen String zu konvertieren. In XPath/XQuery

kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

 © 2014 Altova GmbH

9.2.2 .NET-Erweiterungsfunktionen

Wenn Sie auf einem Windows-Rechner mit der .NET-Plattform arbeiten, können Sie Erweiterungsfunktionen verwenden, die in jeder beliebigen der .NET-Sprachen geschrieben wurden (z.B. C#). Eine .NET Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Konstruktor, eine Eigenschaft oder Methode (statische oder Instanz) in einer .NET-Klasse aufzurufen.

Eine Eigenschaft einer .NET-Klasse wird mit der Syntax `get_PropertyName()` aufgerufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- .NET: Konstruktoren
- .NET: Statische Methoden und statische Felder
- .NET: Instanzmethoden und Instanzfelder
- Datentypen: XPath/XQuery in .NET
- Datentypen: .NET in XPath/XQuery

Form der Erweiterungsfunktion
Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgende Form haben `präfix:fname()`.

- Der Teil `präfix:` ist mit einer URI verknüpft, die die benötigte .NET-Klasse definiert.
- Der Teil `fname()` identifiziert den Konstruktor, die Eigenschaft oder die Methode (statisch oder Instanz) innerhalb der .NET-Klasse und liefert alle gegebenenfalls benötigten Argumente.

- Die URI muss mit `clitype:` beginnen (welches die Funktion als .NET-Erweiterungsfunktion kennzeichnet).
- Die Form `prefix:fname()` der Erweiterungsfunktion kann mit Systemklassen und mit Klasen in einer geladenen Assembly verwendet werden. Wenn eine Klasse allerdings geladen werden muss, müssen zusätzliche Parameter mit den benötigten Informationen bereitgestellt werden.

Parameter
Zum Laden einer Assembly werden die folgenden Parameter verwendet:

`asm`	Der Name der zu ladenden Assembly
`ver`	Die Versionsnummer: eine Maximalzahl von vier Ganzzahlen, die durch Punkte getrennt sind
`sn`	Das Key Token des Strong Name der Assembly (16 Hex-Stellen).
`from`	Eine URI gibt den Pfad der zu ladenden Assembly (DLL) an. Wenn die URI relativ ist, ist sie relativ zum XSLT- oder XQuery-Dokument. Wenn dieser Parameter vorhanden ist, werden alle anderen Parameter ignoriert.
`partialname`	Der partielle Name der Assembly. Er wird für `Assembly.LoadWith.PartialName()` bereitgestellt, welches versuchen wird, die Assembly zu laden. Wenn `partialname` vorhanden ist, werden alle anderen Parameter ignoriert.

loc	Die Locale, z.B. en-US. Die Standardeinstellung ist neutral

Wenn die Assembly aus einer DLL geladen werden soll, verwenden Sie den from Parameter und lassen Sie den sn Parameter weg. Wenn die Assembly aus dem Global Assembly Cache (GAC) geladen werden soll, verwenden Sie den sn Parameter und lassen Sie den from Parameter weg.

Vor dem ersten Parameter muss ein Fragezeichen eingefügt werden. Parameter müssen durch ein Semikolon getrennt werden. Der Wert des Parameternamens wird durch ein Ist-Gleich-Zeichen angegeben (*siehe Beispiele unten*).

Beispiele für Namespace-Deklarationen
Ein Beispiel für eine Namespace Deklaration in XSLT, die die Systemklasse System.Environment: identifiziert.

```
xmlns:myns="clitype:System.Environment"
```

Ein Beispiel für eine Namespace Deklaration in XSLT, die die zu ladende Klasse als Trade.Forward.Scrip: identifiziert.

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Ein Beispiel für eine Namespace-Deklaration in XQuery, die die Systemklasse MyManagedDLL.testClass identifiziert. Es werden zwei Klassen unterschieden:

1. Wenn die Assembly aus dem GAC geladen wird:
```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```

2. Wenn die Assembly aus der DLL geladen wird (vollständige und partielle Referenzen unten):
```
declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/
Altova
Projects/extFunctions/MyManagedDLL.dll;

declare namespace cs="clitype:MyManagedDLL.testClass?
from=MyManagedDLL.dll;
```

XSLT-Beispiel
Hier sehen Sie ein vollständiges XSLT-Beispiel, in dem Funktionen in der Systemklasse System.Math: aufgerufen werden:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
         <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
```

```
      </math>
   </xsl:template>
</xsl:stylesheet>
```

Die Namespace-Deklaration für das Element `math` verknüpft das Präfix `math:` mit der URI `clitype:System.Math`. Der Beginn der URI `clitype:` gibt an, dass danach entweder eine Systemklasse oder eine geladene Klasse definiert wird. Das Präfix `math:` im XPath-Ausdruck verknüpft die Erweiterungsfunktionen mit der URI (und durch Erweiterung der Klasse) `System.Math`. Die Erweiterungsfunktionen identifizieren Methoden in der Klasse `System.Math` und stellen Argumente bereit, wo dies erforderlich ist.

XQuery-Beispiel
Hier sehen Sie ein XQuery-Beispielfragment ähnlich dem XSLT-Beispiel oben:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Wie beim XSLT-Beispiel weiter oben identifiziert die Namespace-Deklaration die .NET-Klasse, in diesem Fall eine Systemklasse. Der XQuery-Ausdruck identifiziert die aufzurufenden Methode und liefert das Argument.

.NET: Konstruktoren

Eine Erweiterungsfunktion kann verwendet werden, um einen .NET-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen. Wenn es mehrere Konstruktoren für eine Klasse gibt, wird der Konstruktor ausgewählt, der der Anzahl der bereitgestellten Argumente am ehesten entspricht. Wenn kein passender Konstruktor gefunden wird, der den bereitgestellten Argumenten entspricht, wird die Fehlermeldung `'No constructor found'` zurückgegeben.

Konstruktoren, die XPath/XQuery-Datentypen zurückgeben
Wenn das Ergebnis eines .NET-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann, gibt die .NET-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einen XPath/XQuery-Datentyp handelt.

Konstruktoren, die .NET-Objekte zurückgeben
Wenn das Ergebnis eines .NET-Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, erstellt der Konstruktor ein wrapped .NET-Objekt mit einem Typ, der der Name der Klasse ist, die dieses Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `System.DateTime` aufgerufen wird (mit `System.DateTime.new()`), so wird ein Objekt mit dem Typ `System.DateTime` zurückgegeben.

Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format eines erforderlichen XPath-Datentyps überein. In solchen Fällen müsste der zurückgegebene Wert: (i) in das lexikalische Format des benötigten XPath-Datentyps konvertiert werden; und (ii) in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes .NET-Objekt kann für drei Zwecke verwendet werden:

- Es kann innerhalb einer Variable verwendet werden:
  ```
  <xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe Instanzmethode und Instanzfelder*):
  ```
  <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann in einen String, eine Zahl oder einen Boolschen Ausdruck konvertiert werden:
- ```
  <xsl:value-of select="xs:integer(data:get_Month(date:new(2008, 4, 29)))"
  xmlns:date="clitype:System.DateTime" />
  ```

.NET: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Namen und durch Angabe der Argumente für die Methode aufgerufen. Der im Aufruf verwendete Name muss exakt mit einer öffentlichen statischen Methode in der angegebenen Klasse übereinstimmen. Wenn der Methodenname und die Anzahl der in der Funktion angegebenen Argumente mit mehr als einer Methode in einer Klasse übereinstimmen, werden die Typen der bereitgestellten Argumente nach der besten Übereinstimmung überprüft. Wenn keine eindeutig passende Methode gefunden werden kann, wird ein Fehler ausgegeben.

Anmerkung: Ein Feld in einer .NET-Klasse wird als Methode ohne Argument betrachtet. Eine Eigenschaft wird mit der Syntax `get_PropertyName()` aufgerufen.

Beispiele

Ein XSLT-Beispiel, in dem Sie einen Methodenaufruf mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf eines Felds (wird als Methode ohne Argument betrachtet) sehen (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()"
xmlns:double="clitype:System.Double"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf einer Eigenschaft (Syntax ist `get_PropertyName()`) (`System.String()`) sehen:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

 © 2014 Altova GmbH
```

Ein XQuery-Beispiel, in dem Sie einen Aufruf einer Methode mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

.NET: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument des Methodenaufrufs ein .NET-Objekt an die Methode übergeben. Dieses .NET-Objekt wird normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. durch einen Konstruktoraufruf) oder einen Stylesheet-Parameter/eine Stylesheet-Variable erstellt. Ein XSLT-Beispiel dieser Art wäre:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
         </date>
         <date>
            <xsl:value-of select="date:ToString($releasedate)"
               xmlns:date="clitype:System.DateTime"/>
         </date>
      </doc>
   </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird ein `System.DateTime` Konstruktor (`new(2008, 4, 29)`) verwendet, um ein .NET-Objekt vom Typ `System.DateTime` zu erstellen. Diese Objekt wird zweimal erstellt, einmal als Wert der Variablen `releasedate`, ein zweites Mal als das erste und einzige Argument der Methode `System.DateTime.ToString()`. Die Instanzmethode `System.DateTime.ToString()` wird zwei Mal aufgerufen, beide Male mit dem `System.DateTime` Konstruktor (`new(2008, 4, 29)`) als erstem und einzigem Argument. In einer dieser Instanzen wird die Variable `releasedate` verwendet, um das .NET-Objekt abzurufen.

Instanzmethoden und Instanzfelder
Der Unterschied zwischen einer Instanzmethode und einem Instanzfeld ist ein theoretischer. In einer Instanzmethode wird ein .NET-Objekt direkt als Argument übergeben; in einem Instanzfeld wird stattdessen ein Parameter oder eine Variable übergeben - auch wenn der Parameter bzw. die Variable selbst ein .NET-Objekt enthalten kann. So enthält z.B. die Variable `releasedate` im

Beispiel oben ein .NET-Objekt und es ist diese Variable, die als das Argument von `ToString()` an den zweiten `date` Elementkonstruktor übergeben wird. Die `ToString()` Instanz im ersten `date` Element ist daher eine Instanzmethode, während die zweite als Instanzfeld betrachtet wird. Das in beiden Instanzen erzeugte Ergebnis ist jedoch dasselbe.

Datentypen: XPath/XQuery in .NET

Wenn in einem XPath/XQuery-Ausdruck eine .NET-Erweiterungsfunktion verwendet wird, spielen die Datentypen der Argumente der Funktion eine wichtige Rolle bei der Entscheidung, welche der vielen .NET-Methoden mit demselben Namen aufgerufen werden soll.

In .NET gelten die folgenden Regeln:

- Wenn es mehr als eine Methode mit demselben Namen in einer Klasse gibt, so stehen nur die Methoden zur Auswahl, die dieselbe Anzahl von Argumenten wie der Funktionsaufruf haben.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden .NET-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen .NET-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener .NET-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene .NET-Methode z.B. `fx(double)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den .NET-Datentyp `double`

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in .NET-Datentypen.

`xs:string`	`StringValue, string`
`xs:boolean`	`BooleanValue, bool`
`xs:integer`	`IntegerValue, decimal, long, integer, short, byte, double, float`
`xs:float`	`FloatValue, float, double`
`xs:double`	`DoubleValue, double`
`xs:decimal`	`DecimalValue, decimal, double, float`

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die .NET-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige .NET-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein `xs:untypedAtomic` Wert 10 und ist für die Methode `mymethod(float)` bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: `mymethod(double)`.
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (`xs:untypedAtomic`) sowohl in `float` als auch `double` korrekt konvertiert werden könnte, kann es geschehen, dass `xs:untypedAtomic` in `double` anstelle von `float` konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode

 © 2014 Altova GmbH

verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. `xs:date`), werden nicht konvertiert und generieren einen Fehler.

Datentypen: .NET in XPath/XQuery

Wenn eine .NET-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird er anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So wird z.B. der .NET-Datentyp `decimal` in `xsd:decimal` konvertiert.

Wenn ein .NET-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine .NET-Methode (z.B. `System.DateTime.ToString()`) verwenden, um das .NET-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

9.2.3 MSXSL-Skripts für XSLT

Das Element <msxsl:script> enthält benutzerdefinierte Funktionen und Variablen, die von XPath-Ausdrücken im XSLT-Stylesheet aufgerufen werden können. Das Element <msxsl:script> ist ein Element der obersten Ebene, d.h. es muss ein Child-Element von <xsl:stylesheet> oder <xsl:transform> sein.

Das Element <msxsl:script> muss sich im Namespace urn:schemas-microsoft-com:xslt (*siehe Beispiel unten*) befinden.

Scripting-Sprache und Namespace
Die im Block verwendete Scripting-Sprache wird im Attribut language des Elements <msxsl:script> definiert und der für Funktionsaufrufe von XPath-Ausdrücken aus zu verwendende Namespace wird durch das Attribut implements-prefix (*siehe unten*) identifiziert.

```
<msxsl:script language="scripting-language implements-prefix="user-namespace-
prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

Das Element <msxsl:script> interagiert mit der Windows Scripting Runtime. Daher können nur Sprachen, die auf Ihrem Rechner installiert sind, im Element <msxsl:script> verwendet werden. **Um MXSL Scripts verwenden zu können muss die Plattform .NET Framework 2.0 oder höher installiert sein.** Folglich können die .NET Scripting Sprachen innerhalb des Elements <msxsl:script> verwendet werden.

Das Attribut language akzeptiert dieselben Werte wie das Attribut language des HTML <script> Elements. Wenn das Attribut language nicht definiert ist, wird als Standardsprache Microsoft JScript verwendet.

Das Attribut implements-prefix erhält einen Wert, der ein Präfix eines deklarierten in-scope Namespace ist. Bei diesem Namespace handelt es sich normalerweise um einen Benutzer-Namespace, der für eine Funktionsbibliothek reserviert ist. Alle Funktionen und Variablen, die im Element <msxsl:script> definiert sind, werden sich im Namespace befinden, der durch das im Attribut implements-prefix definierte Präfixe identifiziert wird. Wenn eine Funktion von einem XPath-Ausdruck aus aufgerufen wird, muss sich der vollständig qualifizierte Funktionsname im selben Namespace wie die Funktionsdefinition befinden.

Beispiel
Hier sehen Sie ein Beispiel für ein vollständiges XSLT Stylesheet, in dem eine Funktion verwendet wird, die in einem <msxsl:script> Element definiert ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input: A currency value: the wholesale price
    ' Returns: The retail price: the input value plus 20% margin,
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Datentypen
Die Werte von Parametern, die an und aus dem Script-Block heraus übergeben werden, sind auf XPath-Datentypen beschränkt. Diese Einschränkung gilt nicht für Daten, die zwischen Funktionen und Variablen innerhalb des Script-Blocks übergeben werden.

Assemblies
Eine Assembly kann über das Element **`msxsl:assembly`** in das Script importiert werden. Die Assembly wird über einen Namen oder eine URL identifiziert. Die Assembly wird beim Kompilieren des Stylesheet importiert. Hier sehen Sie ein einfaches Beispiel, wie das Element `msxsl:assembly` zu verwenden ist.

```
<msxsl:script>
    <msxsl:assembly name="myAssembly.assemblyName" />
    <msxsl:assembly href="pathToAssembly" />
```

```
    ...

</msxsl:script>
```

Der Assembly-Name kann ein vollständiger Name sein, wie z.B.:

```
"system.Math, Version=3.1.4500.1 Culture=neutral
PublicKeyToken=a46b3f648229c514"
```

oder ein Kurzname wie z.B. `"myAssembly.Draw"`.

Namespaces

Namespaces können mit dem Element **`msxsl:using`** deklariert werden. Auf diese Art können Assembly-Klassen ohne ihre Namespaces in das Script geschrieben werden, wodurch Sie sich das mühsame Eintippen ersparen. Hier sehen Sie, wie das Element `msxsl:using` verwendet wird, um Namespaces zu deklarieren.

```
<msxsl:script>
    <msxsl:using namespace="myAssemblyNS.NamespaceName" />

    ...

</msxsl:script>
```

Der Wert des `namespace` Attributs ist der Name des Namespace.

 © 2014 Altova GmbH

Kapitel 10

Altova LicenseServer

10 Altova LicenseServer

Altova LicenseServer (in der Folge auch **LicenseServer** genannt) dient als Tool für die zentrale Verwaltung von Lizenzen für Altova-Produkte. Sie können in einem Netzwerk ausgeführten Altova-Applikationen über LicenseServer eine Lizenz zuweisen. Administratoren können dadurch Lizenzen flexibel verwalten und überwachen.

Aktuelle Version: 1.11

Lizenzierungsverfahren auf Altova LicenseServer
Um einem Altova Server-Produkt über Altova LicenseServer eine Lizenz zuzuweisen, gehen Sie folgendermaßen vor:

1. Starten Sie LicenseServer.
2. Öffnen Sie die LicenseServer-Konfigurationsseite, d.h. die Administratorschnittstelle zu LicenseServer unter Windows, Linux oder Mac OS X.
3. Laden Sie die von Altova erhaltenen Lizenz(en) über das Register "License Pool" Ihrer Altova LicenseServer-Konfigurationsseite in den Lizenzpool.
4. Registrieren Sie das Altova Server-Produkt (FlowForce Server, MapForce Server, StyleVision Server, RaptorXML(+XBRL) Server) auf dem LicenseServer.
5. Weisen Sie dem Altova Server-Produkt auf dem Register "Server Management" der Konfigurationsseite Lizenzen zu.

Die Lizenzen können anschließend über LicenseServer zentral überwacht und verwaltet werden. Eine Beschreibung der verfügbaren Funktionen finden Sie unter Referenz zur Konfigurationsseite.

Anmerkung: Die LicenseServer-Konfigurationsseite unterstützt SSL nicht.

▼ **LicenseServer-Versionen und deren Kompatibilität mit Altova Server-Produkten**
Neue Versionen von Altova Server-Produkten können nur mit der Version von LicenseServer lizenziert werden, die zum Zeitpunkt der Produkt-Release die neueste ist, jedoch funktionieren ältere Versionen von Altova Server-Produkten mit neueren Versionen des LicenseServer.

Wenn Sie daher eine neue Version eines Altova-Produkts installieren und Ihre aktuelle LicenseServer-Version nicht die neueste ist, deinstallieren Sie die ältere Version von LicenseServer und installieren Sie die neueste auf der Altova-Website verfügbare Version. Alle Registrierungs- und Lizenzierungsdaten aus Ihrer alten Version von LicenseServer werden bei der Deinstallation in einer Datenbank auf dem Server gespeichert und automatisch in die neuere Version importiert. Wenn Sie eine neuere Version von LicenseServer installieren, wird die ältere Version vorher deinstalliert.

Die Versionsnummer des aktuell installierten LicenseServer ist am unteren Rand der LicenseServer-Konfigurationsseite (alle Register) vermerkt.

Aktuelle Version: 1.11

 © 2014 Altova GmbH

Informationen zu dieser Dokumentation
Diese Dokumentation ist in die folgenden Teile gegliedert:

- Einführung zu Netzwerkanforderungen; Installation unter Windows, Linux und Mac OS X und Altova ServiceController.
- Zuweisen von Lizenzen: Hier wird Schritt für Schritt beschrieben, wie Sie über den Altova LicenseServer Lizenzen zuweisen.
- Referenz zur Konfigurationsseite: Hier finden Sie eine Beschreibung der Administratorschnittstelle zu LicenseServer.

Letzte Aktualisiereung: 09-15-2014

10.1 Informationen zum Netzwerk

Altova LicenseServer muss auf einem Server-Rechner installiert sein, auf den alle Clients, auf denen lizenzpflichtige Altova-Produkte ausgeführt werden, Zugriff haben. Eine Firewall, sei es auf dem Client als auch auf dem Server, muss den für den ordnungsgemäßen Betrieb des LicenseServers erforderlichen Netzwerkdatenverkehr vom und zum LicenseServer zulassen.

Zur Verteilung der Lizenzen wird auf dem LicenseServer-Rechner **Port 35355** verwendet. Daher muss dieser Port für den Netzwerkdatenverkehr mit Client-Rechnern offen sein.

Im Folgenden sind die Standardparameter für den Netzwerkverkehr und die Anforderungen für LicenseServer aufgelistet:

- *Für die LicenseServer-Lizenzverteilung:*
  Eine oder beide der folgenden Verbindungen
  IPv4 TCP-Verbindung auf Port `35355`
  IPv6 TCP-Verbindung auf Port `35355`

Der LicenseServer kann zu Verwaltungszwecken über eine Webschnittstelle auf Port 8088 aufgerufen werden. Der verwendete Port kann je nach Wunsch konfiguriert werden.

Verbindung zum Master Licensing Server unter altova.com
Altova LicenseServer muss in Verbindung mit dem Master Licensing Server unter `altova.com` stehen, um Lizenzdaten überprüfen und authentifizieren und die Einhaltung der Altova-Lizenzverträge fortlaufend überwachen zu können. Diese Verbindung erfolgt über HTTPS über den Port 443. Wenn der Altova LicenseServer nach dem ersten Herstellen einer Verbindung zum `altova.com` Master Licensing Server länger als 5 Tage (=120 Stunden) keine Verbindung mehr zum Master LicenseServer herstellen kann, lässt der Altova LicenseServer die Verwendung von Altova Software-Produkten, die mit dem Altova LicenseServer in Verbindung stehen, nicht mehr zu.

Wenn die Verbindung zum `altova.com` Master-Server verloren geht, wird dies auf dem Register "Messages" der Konfigurationsseite von LicenseServer protokolliert. Außerdem kann der Administrator den Altova LicenseServer so konfigurieren, dass automatisch eine entsprechende Benachrichtigung gesendet wird, wenn die Verbindung zu `altova.com` unterbrochen wird. Die Benachrichtigungseinstellungen werden auf der Seite "Configuration" auf dem Register "Settings" vorgenommen.

© 2014 Altova GmbH

10.2 Installation (Windows)

Altova LicenseServer kann auf Windows-Systemen auf zwei verschiedene Arten installiert werden:

- mit einem eigenen Installer für Altova LicenseServer
- als Teil des Altova Server-Installationspakets. (Die Altova Server-Produkte sind: Altova FlowForce Server, Altova MapForce Server, Altova StyleVision Server und Altova RaptorXML(+XBRL).)

Falls LicenseServer zu dem Zeitpunkt, zu dem Sie ein Altova Server-Produkt installieren, auf Ihrem System nicht installiert ist, ist die Option zum Installieren von LicenseServer standardmäßig im Installer automatisch ausgewählt. Wenn auf Ihrem Netzwerk bereits ein Altova LicenseServer installiert ist, ist die Option zum Installieren von LicenseServer standardmäßig deaktiviert. Sie können diese Option bei der Installation ändern.

Nähere Informationen zum Zuweisen von Lizenzen finden Sie im Abschnitt Zuweisen von Lizenzen.

LicenseServer-Versionen und deren Kompatibilität mit Altova Server-Produkten

Neue Versionen von Altova Server-Produkten können nur mit der Version von LicenseServer lizenziert werden, die zum Zeitpunkt der Produkt-Release die neueste ist, jedoch funktionieren ältere Versionen von Altova Server-Produkten mit neueren Versionen des LicenseServer.

Wenn Sie daher eine neue Version eines Altova-Produkts installieren und Ihre aktuelle LicenseServer-Version nicht die neueste ist, deinstallieren Sie die ältere Version von LicenseServer und installieren Sie die neueste auf der Altova-Website verfügbare Version. Alle Registrierungs- und Lizenzierungsdaten aus Ihrer alten Version von LicenseServer werden bei der Deinstallation in einer Datenbank auf dem Server gespeichert und automatisch in die neuere Version importiert. Wenn Sie eine neuere Version von LicenseServer installieren, wird die ältere Version vorher deinstalliert.

Die Versionsnummer des aktuell installierten LicenseServer ist am unteren Rand der LicenseServer-Konfigurationseite (alle Register) vermerkt.

Aktuelle Version: 1.11

Welche LicenseServer-Version für eine bestimmte Version eines Server-Produkts benötigt wird, wird während der Installation dieser Version des Server-Produkts angezeigt. Sie können diese Version von LicenseServer zusammen mit dem Server-Produkt installieren oder die neuere Version von LicenseServer separat installieren. In beiden Fällen wird zuerst die ältere Version deinstalliert, bevor die neue Version installiert wird.

10.3 Installation (Linux)

Altova LicenseServer kann auf Linux-Systemen (Debian, Ubuntu, CentOS, RedHat) installiert werden.

Deinstallieren einer alten Version von LicenseServer
Überprüfen Sie in der Linux-Befehlszeile (Command Line Interface = CLI), ob LicenseServer installiert ist. Verwenden Sie dazu den folgenden Befehl:

```
[Debian, Ubuntu]:   dpkg --list | grep Altova
[CentOS, RedHat]:   rpm -qa | grep server
```

Falls LicenseServer nicht installiert ist, beginnen Sie mit der Installation, wie dies in den nächsten Schritten dokumentiert ist. Falls LicenseServer installiert ist und Sie eine neuere Version von LicenseServer installieren möchten, deinstallieren Sie die alte Version mit dem Befehl:

```
[Debian, Ubuntu]:   sudo dpkg --remove licenseserver
[CentOS, RedHat]:   sudo rpm -e licenseserver
```

Installation von Altova LicenseServer
Auf Linux-Systemen muss LicenseServer separat von anderen Altova Server-Produkten installiert werden. LicenseServer bildet nicht Teil der Altova Server-Produktinstallationspakete.
Nachdem Sie das Linux-Paket von der Altova Website heruntergeladen haben, kopieren Sie das Paket in ein beliebiges Verzeichnis auf dem Linux-System.

Distribution	Installer-Erweiterung
Debian	.deb
Ubuntu	.deb
CentOS	.rpm
RedHat	.rpm

Wechseln Sie in einem Terminal-Fenster zu dem Verzeichnis, in das Sie das Linux-Paket kopiert haben. Wenn Sie es z.B. in ein Benutzerverzeichnis namens `MyAltova` (das sich z.B. im Verzeichnis `/home/User` befindet) kopiert haben, dann wechseln Sie folgendermaßen zu diesem Verzeichnis:

```
cd /home/User/MyAltova
```

Installieren Sie LicenseServer mit dem folgenden Befehl:

```
[Debian]:   sudo dpkg --install licenseserver-1.11-debian.deb
[Ubuntu]:   sudo dpkg --install licenseserver-1.11-ubuntu.deb
```

 © 2014 Altova GmbH

```
[CentOS]:   sudo rpm -ivh licenseserver-1.11-1.x86_64.rpm
[RedHat]:   sudo rpm -ivh licenseserver-1.11-1.x86_64.rpm
```

Das LicenseServer-Paket wird im folgenden Ordner installiert:

```
/opt/Altova/LicenseServer
```

Nähere Informationen zum Zuweisen von Lizenzen finden Sie im Abschnitt Zuweisen von Lizenzen.

LicenseServer-Versionen und deren Kompatibilität mit Altova Server-Produkten

Neue Versionen von Altova Server-Produkten können nur mit der Version von LicenseServer lizenziert werden, die zum Zeitpunkt der Produkt-Release die neueste ist, jedoch funktionieren ältere Versionen von Altova Server-Produkten mit neueren Versionen des LicenseServer.

Wenn Sie daher eine neue Version eines Altova-Produkts installieren und Ihre aktuelle LicenseServer-Version nicht die neueste ist, deinstallieren Sie die ältere Version von LicenseServer und installieren Sie die neueste auf der Altova-Website verfügbare Version. Alle Registrierungs- und Lizenzierungsdaten aus Ihrer alten Version von LicenseServer werden bei der Deinstallation in einer Datenbank auf dem Server gespeichert und automatisch in die neuere Version importiert. Wenn Sie eine neuere Version von LicenseServer installieren, wird die ältere Version vorher deinstalliert.

Die Versionsnummer des aktuell installierten LicenseServer ist am unteren Rand der LicenseServer-Konfigurationseite (alle Register) vermerkt.

Aktuelle Version: 1.11

10.4 Installation (Mac OS X)

Altova LicenseServer kann auf Mac OS X Systemen (Version 10.7 oder höher) installiert werden. Da Sie eventuell zuerst eine frühere Version deinstallieren müssen, wird zuerst die Deinstallation von Altova LicenseServer beschrieben.

Deinstallieren alter Versionen von LicenseServer
Bevor Sie LicenseServer deinstallieren, stoppen Sie den Dienst mit dem folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Um zu überprüfen, ob der Dienst gestoppt wurde, öffnen Sie den Activity Monitor Terminal und vergewissern Sie sich, dass sich LicenseServer nicht auf der Liste befindet.

Klicken Sie im Applikations-Terminal mit der rechten Maustaste auf das LicenseServer-Symbol und wählen Sie den Befehl **Move to Trash**. Die Applikation wird daraufhin in den Papierkorb verschoben. Sie müssen die Applikation jetzt noch aus dem Ordner `usr` entfernen. Verwenden Sie dazu den folgenden Befehl:

```
sudo rm -rf /usr/local/Altova/LicenseServer
```

Installieren von Altova LicenseServer
Laden Sie das Altova LicenseServer-Paket von der Altova Website herunter (die Installationsdatei hat die Dateierweiterung `.pkg`) und doppelklicken Sie auf das Installer-Paket, um die Installation zu starten. Befolgen Sie die Anweisungen auf dem Bildschirm. Damit die Installation fortgesetzt wird, müssen Sie die Lizenzvereinbarung akzeptieren.

Das LicenseServer-Paket wird im folgenden Ordner installiert:

```
/usr/local/Altova/LicenseServer
```

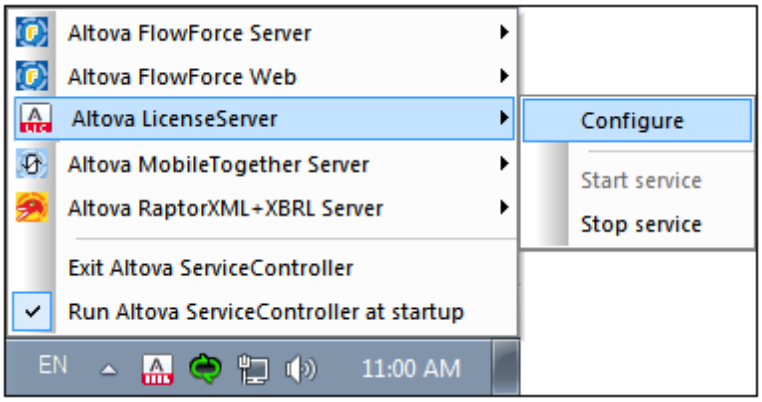
 © 2014 Altova GmbH

10.5 Altova ServiceController

Der Altova ServiceController (in der Folge ServiceController genannt) ist eine Applikation, mit der Sie Altova-Dienste **auf Windows-Systemen** starten, beenden und konfigurieren können.

ServiceController wird mit Altova LicenseServer und jedem *Altova Server-Produkt, das ServiceController verwendet* FlowForce Server, RaptorXML(+XBRL) Server und Mobile Together Server, installiert und kann durch Klicken auf **Start | Altova LicenseServer | Altova ServiceController** gestartet werden. (Dieser Befehl steht auch in den **Startmenüordnern** der *Altova Server-Produkten, die als Dienst installiert wurden* (FlowForce Server, RaptorXML(+XBRL) Server und Mobile Together Server), zur Verfügung). Nachdem der ServiceController gestartet wurde, kann er über die Task-Leiste aufgerufen werden (*Abbildung unten*).

Damit der ServiceController automatisch ausgeführt wird, nachdem sich der Benutzer im System angemeldet hat, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol, um das **ServiceController**-Menü (*Abbildung unten*) aufzurufen und aktivieren Sie anschließend den Befehl **Run Altova ServiceController at Startup**. (Dieser Befehl ist standardmäßig aktiv). Um den ServiceController zu beenden, klicken Sie in der Task-Leiste auf das **ServiceController**-Symbol und klicken Sie im Menü, das daraufhin angezeigt wird (*Abbildung unten*) auf **Exit Altova ServiceController**.



Starten und Beenden von Altova-Diensten
Für jede installierte Altova Dienstkomponente gibt es im Menü "ServiceController" (*siehe Abbildung oben*) einen Eintrag. Ein Altova-Dienst kann über einen Befehl im ServiceController-Untermenü gestartet oder beendet werden. Außerdem können über das Menü "ServiceController" wichtige Verwaltungsaufgabe einzelner Dienste aufgerufen werden. So hat z.B. der Dienst Altova LicenseServer in der Abbildung oben ein Untermenü **Configure**, über das Sie die Konfigurationsseite von LicenseServer aufrufen können.

10.6 Zuweisen von Lizenzen

Um einem Altova Server-Produkt über den Altova LicenseServer eine Lizenz zuzuweisen, gehen Sie folgendermaßen vor:

1. Starten Sie LicenseServer.
2. Öffnen Sie die LicenseServer-Konfigurationsseite, d.h. die Administratorschnittstelle zu LicenseServer unter Windows, Linux oder Mac OS X.
3. Laden Sie die von Altova erhaltenen Lizenz(en) über das Register "License Pool" Ihres Altova LicenseServers in den Lizenzpool.
4. Registrieren Sie das Altova Server-Produkt (FlowForce Server, MapForce Server, StyleVision Server) auf dem LicenseServer. Je nach Produkttyp erfolgt die Registrierung beim LicenseServer auf unterschiedliche Art: entweder über die Web-Benutzeroberfläche des Produkts oder über seine Befehlszeile. Nähere Informationen finden Sie in der Dokumentation zu Ihrem Altova Server-Produkt.
5. Weisen Sie dem Altova Server-Produkt auf dem Register "Server Management" der LicenseServer-Konfigurationsseite eine Lizenz für die entsprechende Anzahl von Prozessorkernen auf dem Rechner zu.

Anmerkung zu Prozessorkernen und Lizenzen
Die Lizenzierung von Altova Server-Produkten basiert *mit Ausnahme von MobileTogether Server****
auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor hat vier Kerne, ein Hexa-Core-Prozessor hat sechs Kerne, usw. Die Anzahl der für ein Produkt auf einem bestimmten Server-Rechner lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Server verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt.

Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie eine Lizenz für mindestens acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen Octa-Core-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit eine großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Lizenz kann immer nur für einen Client gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität noch nicht ausgeschöpft ist. Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

*** MobileTogether Server-Lizenzen werden auf Basis der Anzahl der Benutzer, d.h. auf Basis der Clients, die auf MobileTogether Server zugreifen, zugewiesen.

 © 2014 Altova GmbH

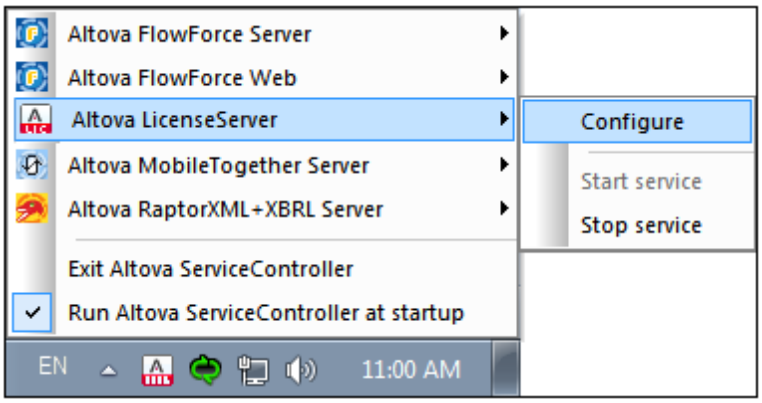
10.6.1 Starten von LicenseServer

In diesem Abschnitt werden folgende Schritte beschrieben:

- Starten von LicenseServer auf Windows-Systemen
- Starten von LicenseServer auf Linux-Systemen
- Starten von LicenseServer auf Mac OS X-Systemen
- Anmerkung zur Verbindung zu altova.com

Windows-Systeme
Sie können LicenseServer über den Altova ServiceController, der in der Task-Leiste zur Verfügung steht, starten.

Klicken Sie zuerst auf **Start | Alle Programme | Altova LicenseServer | Altova ServiceController,** um Altova ServiceController zu starten und sein Symbol in der Task-Leiste anzuzeigen (*siehe Abbildung unten*). Falls Sie die Option *Run Altova ServiceController at Startup* aktivieren, wird Altova ServiceController beim Systemstart gestartet, sodass sein Symbol ab diesem Zeitpunkt in der Task-Leiste zur Verfügung steht.



Um LicenseServer zu starten, klicken Sie in der Task-Leiste auf das Altova ServiceController-Symbol und platzieren Sie den Mauszeiger im angezeigten Menü über **Altova LicenseServer** (*Abbildung oben*) und wählen Sie im Untermenü von Altova LicenseServer den Befehl **Start Service**. Wenn LicenseServer bereits ausgeführt wird, ist die Option *Start Service* deaktiviert.

Linux-Systeme
Um LicenseServer auf Linux-Systemen als Dienst zu starten, führen Sie im Terminal-Fenster den folgenden Befehl aus.

```
[Debian]:          sudo /etc/init.d/licenseserver start

 [Ubuntu]:          sudo initctl start licenseserver

 [CentOS]:          sudo initctl start licenseserver
```

```
[RedHat]:          sudo initctl start licenseserver
```

(Wenn Sie LicenseServer beenden möchten, ersetzen Sie den Befehl **start** im obigen Befehl durch **stop**.)

Mac OS X-Systeme
Um LicenseServer auf Mac OS X-Systemen als Dienst zu starten, führen Sie in einem Terminal-Fenster den folgenden Befehl aus:

```
sudo launchctl load /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Um LicenseServer zu beenden, verwenden Sie den folgenden Befehl:

```
sudo launchctl unload /Library/LaunchDaemons/com.altova.LicenseServer.plist
```

Verbindung zum Master Licensing Server unter altova.com
Altova LicenseServer muss in Verbindung mit dem Master Licensing Server unter `altova.com` stehen, um Lizenzdaten überprüfen und authentifizieren und die Einhaltung der Altova-Lizenzverträge fortlaufend überwachen zu können. Diese Verbindung erfolgt über HTTPS über den Port 443. Wenn der Altova LicenseServer nach dem ersten Herstellen einer Verbindung zum `altova.com` Master Licensing Server länger als 5 Tage (=120 Stunden) keine Verbindung mehr zum Master LicenseServer herstellen kann, lässt der Altova LicenseServer die Verwendung von Altova Software-Produkten, die mit dem Altova LicenseServer in Verbindung stehen, nicht mehr zu.

Wenn die Verbindung zum `altova.com` Master-Server verloren geht, wird dies auf dem Register "Messages" der Konfigurationsseite von LicenseServer protokolliert. Außerdem kann der Administrator den Altova LicenseServer so konfigurieren, dass automatisch eine entsprechende Benachrichtigung gesendet wird, wenn die Verbindung zu `altova.com` unterbrochen wird. Die Benachrichtigungseinstellungen werden auf der Seite "Configuration" auf dem Register "Settings" vorgenommen.

© 2014 Altova GmbH

10.6.2 Öffnen der LicenseServer-Konfigurationsseite (Windows)

In diesem Abschnitt werden folgende Schritte beschrieben:

- Öffnen der Konfigurationsseite, wenn LicenseServer auf demselben Rechner installiert ist
- Öffnen der Konfigurationsseite, wenn LicenseServer auf einem anderen Rechner installiert ist
- Einloggen mit dem Anfangspasswort
- Definieren eines festgelegten Ports für die Konfigurationsseite

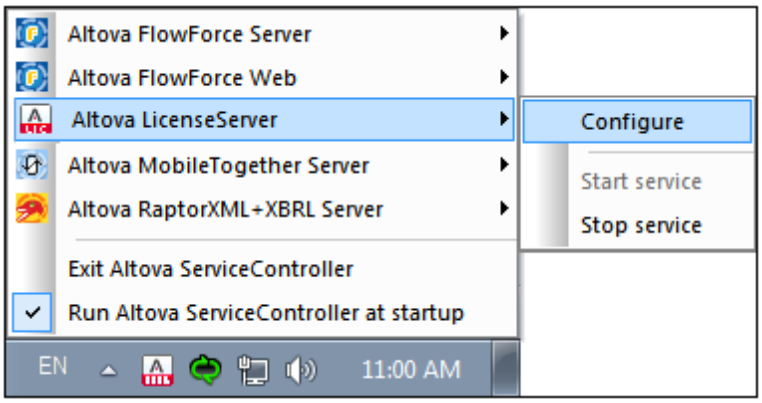
---

Öffnen der Konfigurationsseite, wenn LicenseServer auf demselben Rechner installiert ist

Auf Windows-Systemen können Sie die Konfigurationsseite von LicenseServer auf zwei Arten aufrufen, wenn LicenseServer auf demselben Rechner installiert ist:

- Klicken Sie auf **Start | Alle Programme | Altova LicenseServer | LicenseServer-Konfigurationsseite**. Daraufhin wird die Konfigurationsseite auf einem neuen Register Ihres Internet Browsers geöffnet.

- Klicken Sie in der Task-Leiste auf das Altova ServiceController-Symbol und platzieren Sie den Mauszeiger im angezeigten Menü über **Altova LicenseServer** (*Abbildung unten*) und wählen Sie im Untermenü von Altova LicenseServer den Befehl **Configure**.



Daraufhin wird die Konfigurationsseite in einem neuen Browser-Fenster geöffnet und die Login-Maske wird angezeigt (*siehe Abbildung unten*).

Öffnen der Konfigurationsseite, wenn LicenseServer auf einem anderen Rechner installiert ist

Um die Konfigurationsseite von LicenseServer von einem anderen Windows-Rechner im lokalen Netzwerk (als dem, auf dem LicenseServer installiert ist) zu öffnen, geben Sie die URL der LicenseServer Konfigurationsseite in die Adressleiste des Browsers ein und drücken Sie die **Eingabetaste**. Standardmäßig lautet die URL der Konfigurationsseite:

```
http://<serverIPAddressOrName>:8088/
```

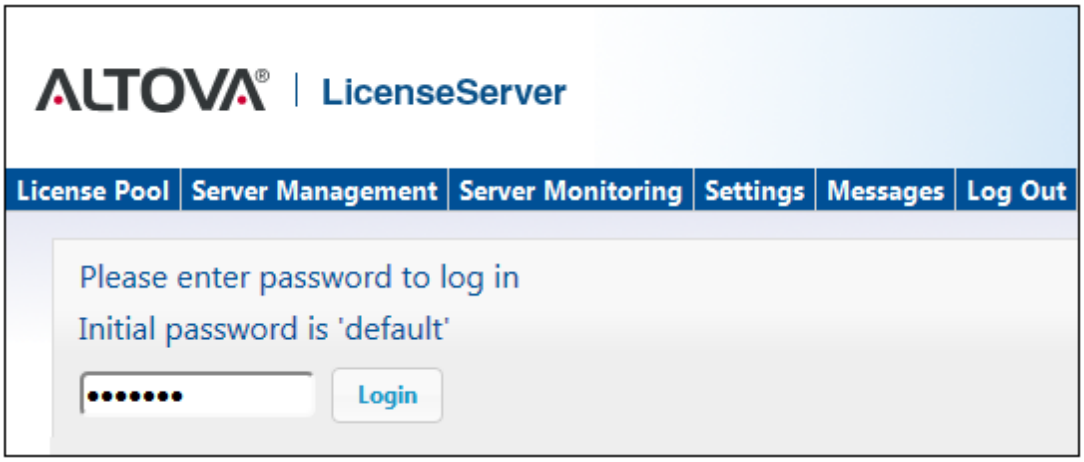
Die URL ist im HTML-Code der Konfigurationsseite selbst enthalten. Der Name der Konfigurationsseite lautet `WebUI.html` und befindet sich unter:

```
C:/ProgramData/Altova/LicenseServer/WebUI.html
```

Wenn Sie (auf der Konfigurationsseite auf dem Register "Settings") für die URL der Konfigurationsseite eingestellt haben, dass sie dynamisch generiert wird, so wird bei jedem Start von LicenseServer eine neue URL generiert. Sie müssen die aktuelle Version von **WebUI.html** überprüfen, um die aktuelle URL der Konfigurationsseite zu ermitteln.

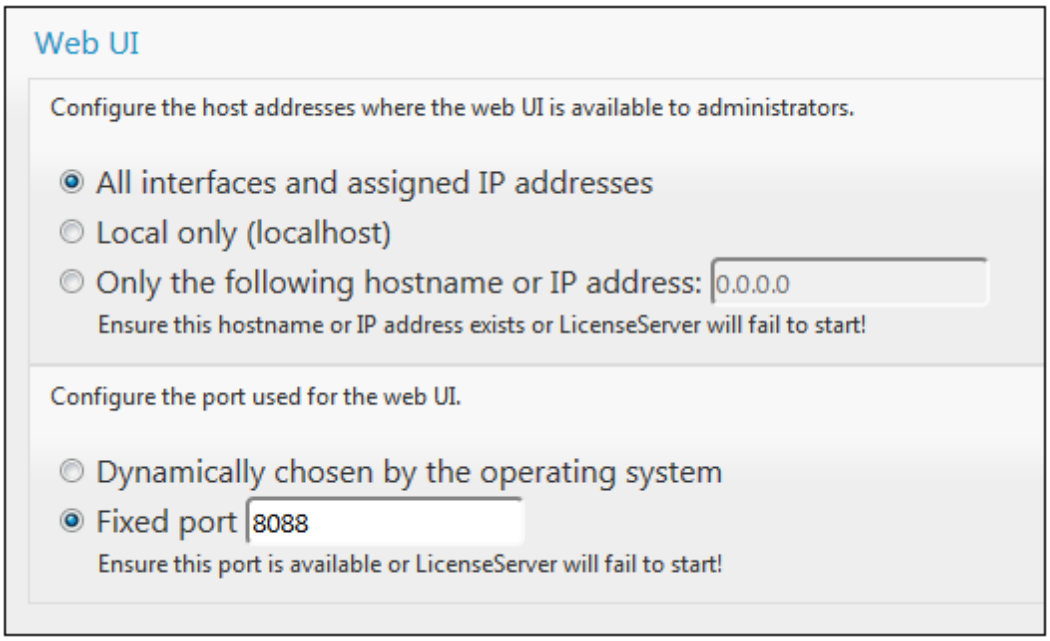
Die dynamisch in **WebUI.html** generierte URL sollte in etwa folgendermaßen lauten: `http://127.0.0.1:55541/optional-ein-weiterer-String` und befindet sich in der Funktion `checkIfServiceRunning()` in einem Script nahe dem Ende des Elements `<head>`. Während die Port-Nummer in der URL dynamisch zugewiesen wird, identifiziert die IP-Adresse in der URL den Server, auf dem LicenseServer installiert wurde. Wenn Sie die LicenseServer-Konfigurationsseite von einem anderen Rechner aus aufrufen möchten, stellen Sie sicher, dass der IP-Adressteil der URL die richtige IP-Adresse oder den richtigen Namen des Servers hat, auf dem LicenseServer installiert wurde. Die URL könnte z.B. lauten: `http://MyServer:55541`.

Einloggen mit dem Anfangspasswort
Nachdem Sie die oben beschriebenen Schritte durchgeführt haben, wird die Konfigurationsseite mit der Login-Maske angezeigt (*Abbildung unten*). Sie können sich mit dem vorgegebenen Anfangspasswort `default` einloggen. Sie können Ihr Passwort nach dem Einloggen auf dem Register Settings ändern.



Definieren eines fixen oder dynamischen Ports für die Konfigurationsseite
Der Port für die Konfigurationsseite (Web-Benutzeroberfläche) (und somit seine Adresse) kann auf der Seite "Settings" definiert werden. Standardmäßig lautet der Port `8088`. Sie können auf der Konfigurationsseite (*siehe Abbildung unten*) für den LicenseServer jeden beliebigen anderen gewünschten Port definieren. Alternativ dazu können Sie auch festlegen, dass der Port bei jedem Start von LicenseServer dynamisch gewählt wird. In diesem Fall müssen Sie die URL der Konfigurationsseite in der Datei `WebUI.html` nachsehen (siehe Öffnen der LicenseServer-Konfigurationsseite (Windows) und Öffnen der LicenseServer-Konfigurationsseite (Linux)..

 © 2014 Altova GmbH


Web UI
Configure the host addresses where the web UI is available to administrators.
All interfaces and assigned IP addresses
Local only (localhost)
Only the following hostname or IP address:
0.0.0.0
Ensure this hostname or IP address exists or LicenseServer will fail to start!
Configure the port used for the web UI.
Dynamically chosen by the operating system
Fixed port
8088
Ensure this port is available or LicenseServer will fail to start!

Der Vorteil eines fixen Ports ist, dass die Seiten-URL im Vorhinein bekannt ist und daher problemlos aufgerufen werden kann. Wenn der Port dynamisch zugewiesen wird, muss der Port der URL bei jedem Start von LicenseServer zuerst in der Datei `WebUI.html` nachgeschlagen werden.

10.6.3 Öffnen der LicenseServer-Konfigurationsseite (Linux)

In diesem Abschnitt werden folgende Schritte beschrieben:

- Erstmaliges Öffnen der Konfigurationsseite über die zurückgegebene URL
- URL der LicenseServer-Konfigurationsseite
- Einloggen mit dem Anfangspasswort
- Definieren eines festgelegten Ports für die Konfigurationsseite

---

Erstmaliges Öffnen der Konfigurationsseite über die zurückgegebene URL
Wenn Sie Ihr Altova Server-Produkt auf Linux-Systemen über das CLI am LicenseServer registrieren, wird die URL der LicenseServer-Konfigurationsseite zurückgegeben. Wenn Sie diese URL in einem Browser öffnen, werden Sie aufgefordert, den Lizenzvertrag zu lesen und den Bedingungen zuzustimmen. Nachdem Sie Ihre Zustimmung gegeben haben, wird die Login-Maske der Konfigurationsseite angezeigt (*Abbildung unten*).

URL der LicenseServer-Konfigurationsseite
Um die Konfigurationsseite von LicenseServer jederzeit aufrufen zu können, geben Sie die URL der LicenseServer Konfigurationsseite in die Adressleiste des Browsers ein und drücken Sie die **Eingabetaste**. Standardmäßig lautet die URL der Konfigurationsseite:

```
http://<serverIPAddressOrName>:8088/
```

Die URL ist im HTML-Code der Konfigurationsseite selbst enthalten. Der Name der Konfigurationsseite lautet `WebUI.html` und befindet sich unter:

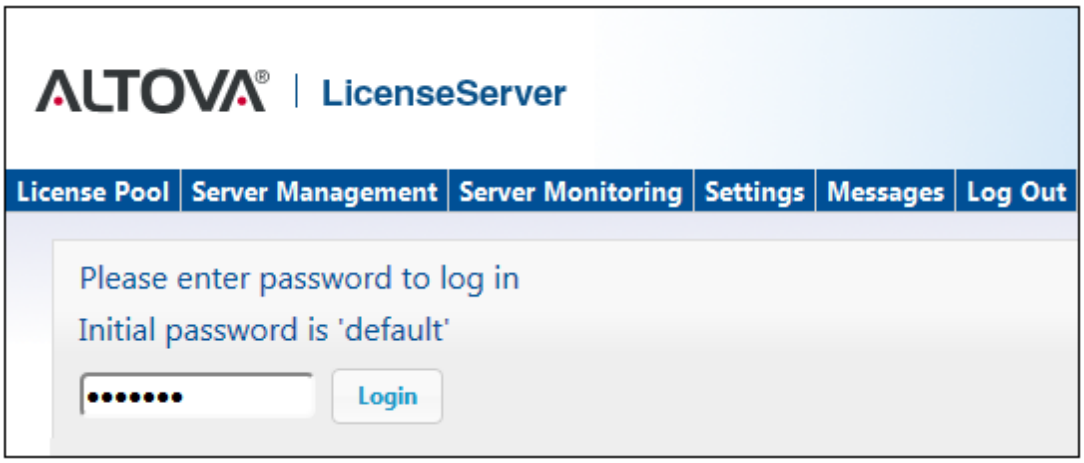
```
/var/opt/Altova/LicenseServer/webUI.html
```

Wenn Sie (auf der Konfigurationsseite auf dem Register "Settings") für die URL der Konfigurationsseite eingestellt haben, dass sie dynamisch generiert wird, so wird bei jedem Start von LicenseServer eine neue URL generiert. Sie müssen die aktuelle Version von `WebUI.html` überprüfen, um die aktuelle URL der Konfigurationsseite zu ermitteln.

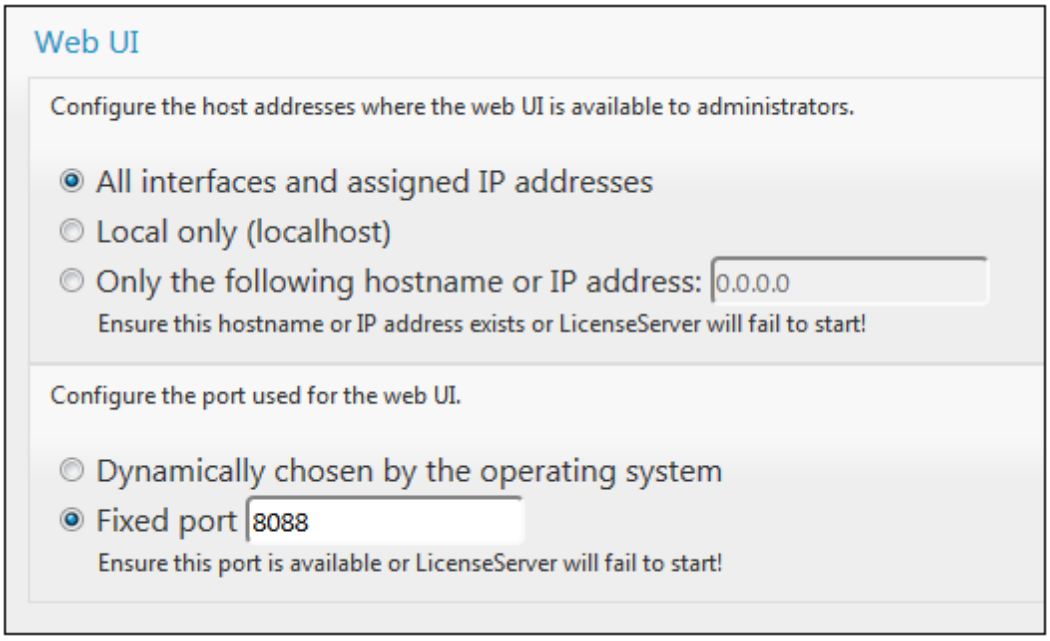
Die dynamisch in `WebUI.html` generierte URL sollte in etwa folgendermaßen lauten: `http://127.0.0.1:55541` und befindet sich in der Funktion `checkIfServiceRunning()` in einem Script nahe dem Ende des Elements `<head>`. Während die Port-Nummer in der URL dynamisch zugewiesen wird, identifiziert die IP-Adresse in der URL den Server, auf dem LicenseServer installiert wurde. Wenn Sie die LicenseServer-Konfigurationsseite von einem anderen Rechner aus aufrufen möchten, stellen Sie sicher, dass der IP-Adressteil der URL die richtige IP-Adresse oder den richtigen Namen des Servers hat, auf dem LicenseServer installiert wurde. Die URL könnte z.B. lauten: `http://MyServer:55541`.

Einloggen mit dem Anfangspasswort
Nachdem Sie die oben beschriebenen Schritte durchgeführt haben, wird die Konfigurationsseite

mit der Login-Maske angezeigt (*Abbildung unten*). Sie können sich mit dem vorgegebenen Anfangspasswort `default` einloggen. Sie können Ihr Passwort nach dem Einloggen auf dem Register Settings ändern.



Definieren eines fixen oder dynamischen Ports für die Konfigurationsseite
Der Port für die Konfigurationsseite (Web-Benutzeroberfläche) (und somit seine Adresse) kann auf der Seite "Settings" definiert werden. Standardmäßig lautet der Port `8088`. Sie können auf der Konfigurationsseite (*siehe Abbildung unten*) für den LicenseServer jeden beliebigen anderen gewünschten Port definieren. Alternativ dazu können Sie auch festlegen, dass der Port bei jedem Start von LicenseServer dynamisch gewählt wird. In diesem Fall müssen Sie die URL der Konfigurationsseite in der Datei `WebUI.html` nachsehen (siehe Öffnen der LicenseServer-Konfigurationsseite (Windows) und Öffnen der LicenseServer-Konfigurationsseite (Linux)..


Web UI
Configure the host addresses where the web UI is available to administrators.
All interfaces and assigned IP addresses
Local only (localhost)
Only the following hostname or IP address: 0.0.0.0
Ensure this hostname or IP address exists or LicenseServer will fail to start!
Configure the port used for the web UI.
Dynamically chosen by the operating system
Fixed port 8088
Ensure this port is available or LicenseServer will fail to start!

Der Vorteil eines fixen Ports ist, dass die Seiten-URL im Vorhinein bekannt ist und daher problemlos aufgerufen werden kann. Wenn der Port dynamisch zugewiesen wird, muss der Port der URL bei jedem Start von LicenseServer zuerst in der Datei `WebUI.html` nachgeschlagen werden.

 © 2014 Altova GmbH

10.6.4 Öffnen der LicenseServer-Konfigurationsseite (Mac OS X)

In diesem Abschnitt werden folgende Schritte beschrieben:

- Erstmaliges Öffnen der Konfigurationsseite über die zurückgegebene URL
- URL der LicenseServer-Konfigurationsseite
- Einloggen mit dem Anfangspasswort
- Definieren eines festgelegten Ports für die Konfigurationsseite

---

Erstmaliges Öffnen der Konfigurationsseite über die zurückgegebene URL
Wenn Sie Ihr Altova Server-Produkt auf Mac OS X-Systemen über das CLI am LicenseServer registrieren, wird die URL der LicenseServer-Konfigurationsseite zurückgegeben. Wenn Sie diese URL in einem Browser öffnen, werden Sie aufgefordert, den Lizenzvertrag zu lesen und den Bedingungen zuzustimmen. Nachdem Sie Ihre Zustimmung gegeben haben, wird die Login-Maske der Konfigurationsseite angezeigt (*Abbildung unten*).

URL der LicenseServer-Konfigurationsseite
Um die Konfigurationsseite von LicenseServer jederzeit aufrufen zu können, geben Sie die URL der LicenseServer Konfigurationsseite in die Adressleiste des Browsers ein und drücken Sie die **Eingabetaste**. Standardmäßig lautet die URL der Konfigurationsseite:

```
http://<serverIPAddressOrName>:8088/
```

Die URL ist im HTML-Code der Konfigurationsseite selbst enthalten. Der Name der Konfigurationsseite lautet `WebUI.html` und befindet sich unter:

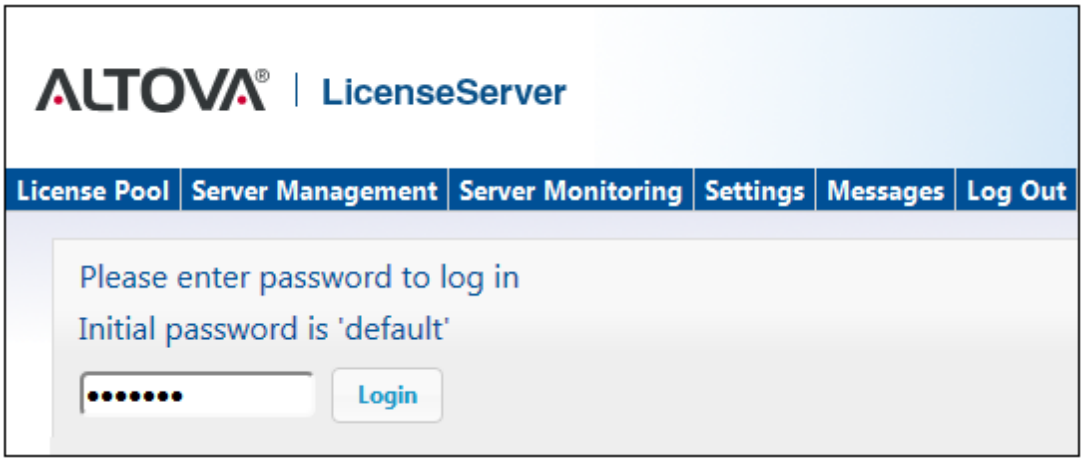
```
/var/opt/Altova/LicenseServer/webUI.html
```

Wenn Sie (auf der Konfigurationsseite auf dem Register "Settings") für die URL der Konfigurationsseite eingestellt haben, dass sie dynamisch generiert wird, so wird bei jedem Start von LicenseServer eine neue URL generiert. Sie müssen die aktuelle Version von `WebUI.html` überprüfen, um die aktuelle URL der Konfigurationsseite zu ermitteln.

Die dynamisch in `WebUI.html` generierte URL sollte in etwa folgendermaßen lauten: `http://127.0.0.1:55541` und befindet sich in der Funktion `checkIfServiceRunning()` in einem Script nahe dem Ende des Elements `<head>`. Während die Port-Nummer in der URL dynamisch zugewiesen wird, identifiziert die IP-Adresse in der URL den Server, auf dem LicenseServer installiert wurde. Wenn Sie die LicenseServer-Konfigurationsseite von einem anderen Rechner aus aufrufen möchten, stellen Sie sicher, dass der IP-Adressteil der URL die richtige IP-Adresse oder den richtigen Namen des Servers hat, auf dem LicenseServer installiert wurde. Die URL könnte z.B. lauten: `http://MyServer:55541`.

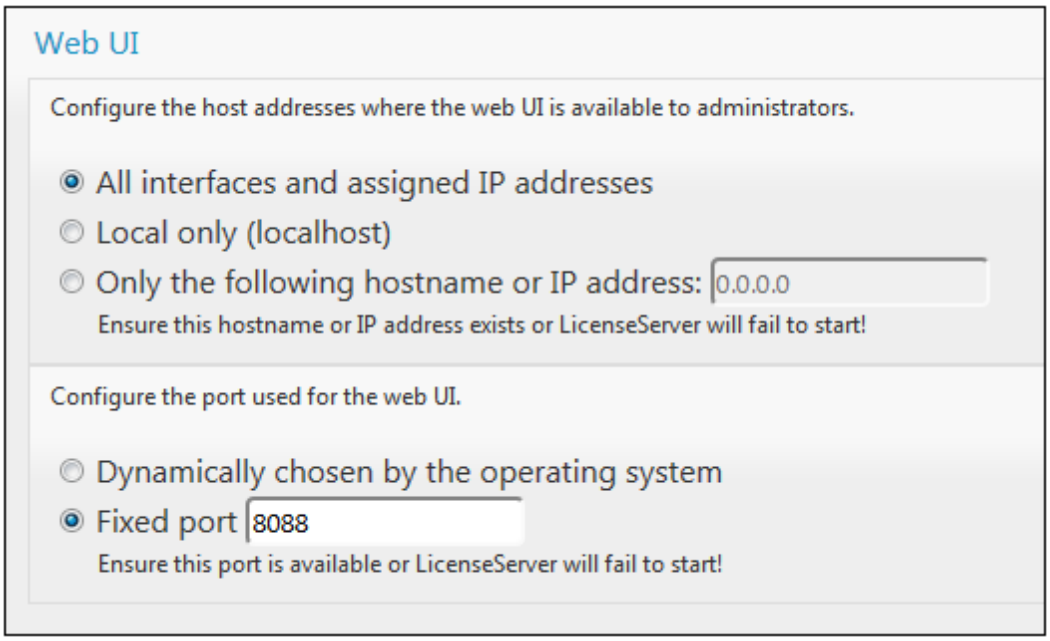
Anmerkung: Die Konfigurationsseite kann auch direkt über das Symbol **Suche | Anwendungen | Altova License Server** aufgerufen werden.

Einloggen mit dem Anfangspasswort
Nachdem Sie die oben beschriebenen Schritte durchgeführt haben, wird die Konfigurationsseite mit der Login-Maske angezeigt (*Abbildung unten*). Sie können sich mit dem vorgegebenen Anfangspasswort `default` einloggen. Sie können Ihr Passwort nach dem Einloggen auf dem Register Settings ändern.



Definieren eines fixen oder dynamischen Ports für die Konfigurationsseite
Der Port für die Konfigurationsseite (Web-Benutzeroberfläche) (und somit seine Adresse) kann auf der Seite "Settings" definiert werden. Standardmäßig lautet der Port `8088`. Sie können auf der Konfigurationsseite (*siehe Abbildung unten*) für den LicenseServer jeden beliebigen anderen gewünschten Port definieren. Alternativ dazu können Sie auch festlegen, dass der Port bei jedem Start von LicenseServer dynamisch gewählt wird. In diesem Fall müssen Sie die URL der Konfigurationsseite in der Datei `WebUI.html` nachsehen (siehe Öffnen der LicenseServer-Konfigurationsseite (Windows) und Öffnen der LicenseServer-Konfigurationsseite (Linux)..

 © 2014 Altova GmbH


Web UI
Configure the host addresses where the web UI is available to administrators.
All interfaces and assigned IP addresses
Local only (localhost)
Only the following hostname or IP address:
0.0.0.0
Ensure this hostname or IP address exists or LicenseServer will fail to start!
Configure the port used for the web UI.
Dynamically chosen by the operating system
Fixed port
8088
Ensure this port is available or LicenseServer will fail to start!

Der Vorteil eines fixen Ports ist, dass die Seiten-URL im Vorhinein bekannt ist und daher problemlos aufgerufen werden kann. Wenn der Port dynamisch zugewiesen wird, muss der Port der URL bei jedem Start von LicenseServer zuerst in der Datei `WebUI.html` nachgeschlagen werden.

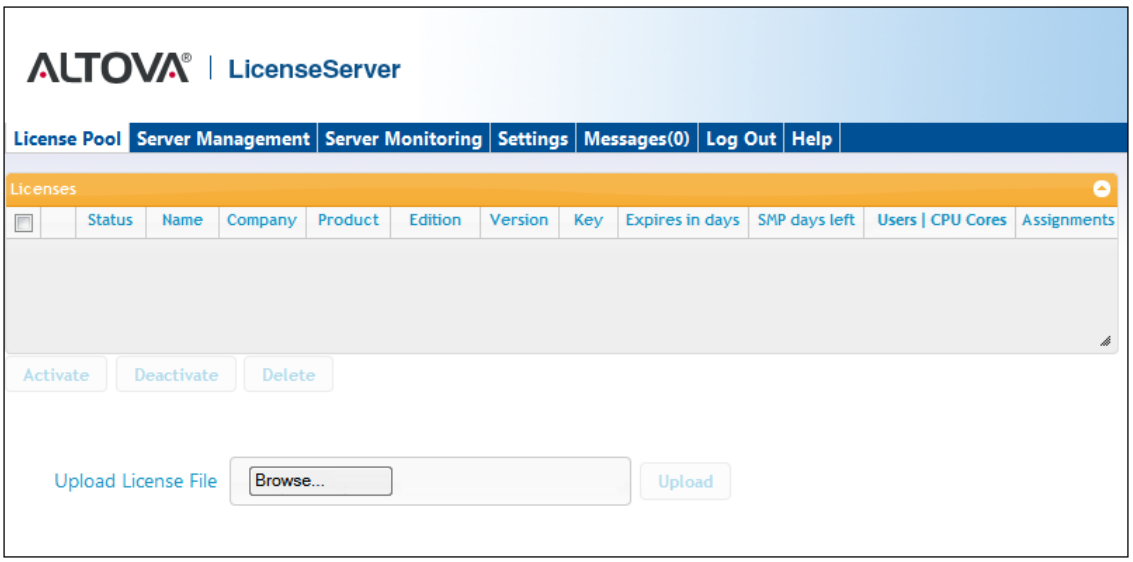
10.6.5 Laden von Lizenzen in LicenseServer

In diesem Abschnitt werden folgende Schritte beschrieben:

- Laden einer Lizenzdatei in den Lizenzpool von LicenseServer
- Lizenzstatus
- Aktivieren der gewünschten Lizenz
- Nächste Schritte

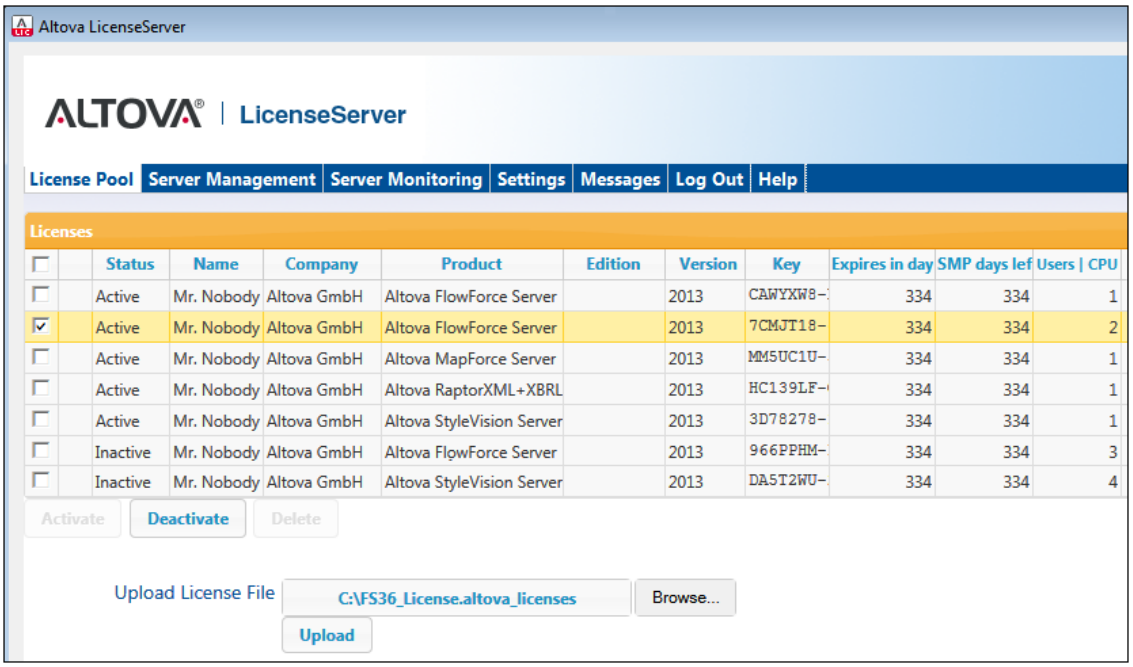
Laden einer Lizenzdatei in den Lizenzpool von LicenseServer
Nachdem Sie von Altova eine Lizenzdatei erhalten haben, müssen Sie diese im Altova LicenseServer laden. (Beschreibung siehe unten). Jede Lizenzdatei kann je nach Kauf eine oder mehrere Lizenzen enthalten. Wenn Sie eine Lizenzdatei laden, werden alle darin enthaltenen Lizenzen am Server geladen und können einem bei diesem LicenseServer registrierten Altova-Produkt zugewiesen werden. Alle geladenen Lizenzen aus einer oder mehreren Lizenzdateien und für alle Altova-Produkte werden am LicenseServer in einem Lizenz-Pool gesammelt. Der Lizenz-Pool wird auf der LicenseServer-Konfigurationsseite auf dem Register "License Pool" (*Abbildung unten*) angezeigt.

Lizenzdateien werden über die Funktion "Upload" auf dem Register "License Pool" (*siehe Abbildung unten*) in den LicenseServer geladen.



Klicken Sie auf die Schaltfläche **Browse** und wählen Sie die gewünschte Lizenzdatei aus. Der Lizenzdateiname wird im Textfeld "Upload License File" angezeigt und die Schaltfläche **Upload** wird aktiv. Klicken Sie auf die Schaltfläche **Upload,** um die Lizenzdatei zu laden. Alle Lizenzen in der Datei werden geladen und auf dem Register "License Pool" angezeigt. In der Abbildung unten sehen Sie mehrere Lizenzen, die aus mehreren Lizenzdateien geladen wurden.

© 2014 Altova GmbH



Lizenzstatus
Es gibt die folgenden Lizenzstatuswerte:

- *Activating:* Nachdem eine Lizenz in den Lizenzpool von LicenseServer hochgeladen wurde, werden die Lizenzdaten dazu zur Überprüfung, Authentifizierung und Aktivierung der geladenen Lizenz an den Master Licensing Server unter `altova.com` weitergeleitet, um die Einhaltung des Altova Lizenzvertrags zu überwachen. Währenddessen - die Dauer kann je nach Verbindung und Netzwerkverkehr zwischen 30 Sekunden und einigen Minuten variieren - wird der Status als *Activating....*angezeigt.

- *Failed Verification:* Falls keine Verbindung zum Master Licensing Server unter `altova.com` hergestellt werden konnte, wird der Status der Lizenz im Pool als *Failed Verification* angezeigt. Überprüfen Sie in diesem Fall Ihre Internetverbindung und die Firewall-Einstellungen, um sicherzustellen, dass LicenseServer eine Verbindung zum Master Licensing Server unter `altova.com` herstellen kann.

- *Active:* Sobald die Lizenz authentifiziert und aktiviert wurde, ändert sich der Status im Pool in *Active.*

- *Inactive:* Wenn eine Lizenz überprüft wurde, aber auf einem anderen LicenseServer im Netzwerk vorhanden ist, wird der Status im Pool als *Inactive* angezeigt. Der Status *Inactive* kann auch zustande kommen, wenn eine Lizenz im Lizenzpool vom Administrator manuell deaktiviert wurde.

- *Blocked:* Eine Lizenz erscheint im Lizenzpool als *Blocked,* wenn es ein Problem mit der Authentifizierung der Lizenz gab und der Master License Server unter `altova.com` die Verwendung der Lizenz nicht gestattet. Dies kann der Fall sein, wenn der Lizenzvertrag nicht eingehalten wurde, zu viele Lizenzen verwendet werden oder andere

Lizenzverletzungen festgestellt wurden. Wird eine Lizenz mit dem Status *Blocked* angezeigt, wenden Sie sich bitte mit Ihren Lizenzdaten und anderen relevanten Daten an den Altova Support.

Eine Übersicht über die Statuswerte finden Sie in der folgenden Tabelle:

Status	Bedeutung
Activating...	Nach dem Laden werden die Lizenzdaten zur Überprüfung an `altova.com` gesendet. Aktualisieren Sie den Browser, um den aktualisierten Status zu sehen. Die Überprüfung und Aktivierung kann einige Minuten dauern.
Failed Verification	Es konnte keine Verbindung zu `altova.com` hergestellt werden. Nachdem eine Verbindung hergestellt wurde, starten Sie den Dienst entweder neu oder aktivieren Sie die Lizenz (über die Schaltfläche **Activate**).
Active	Die Überprüfung war erfolgreich, die Lizenz ist aktiv.
Inactive	Die Überprüfung war erfolgreich, aber die Lizenz wird derzeit auf einem anderen LicenseServer im Netzwerk verwendet. Lizenzen können über die Schaltfläche **Deactivate** deaktiviert werden.
Blocked	Die Überprüfung war negativ. Die Lizenz ist ungültig und wurde blockiert. Wenden Sie sich an den Altova Support

Anmerkung: Nachdem die Lizenz zur Überprüfung an `altova.com` gesendet wurde, muss der Browser aktualisiert werden, damit Sie den aktualisierten Status sehen. Die Überprüfung und Aktivierung kann einige Minuten dauern.

Anmerkung: Falls keine Verbindung zu `altova.com` hergestellt werden konnte, wird der Status als *Failed Verification* angezeigt. Nachdem eine Verbindung hergestellt wurde, starten Sie den Dienst entweder neu oder versuchen Sie, die Lizenz (über die Schaltfläche **Activate**) zu aktivieren.

Anmerkung: Wenn eine Lizenz den Status *Inactive* oder *Blocked* erhält, wird eine Meldung mit einer Erklärung dazu zum Meldungs-Log hinzugefügt.

Nur eine aktive Lizenz kann einem Produkt zugewiesen werden. Eine inaktive Lizenz kann aktiviert oder aus dem Lizenz-Pool gelöscht werden. Wenn eine Lizenz aus dem Lizenz-Pool gelöscht wurde, kann sie erneut durch Laden der darin enthaltenen Lizenzdatei in den Pool geladen werden. Wenn eine Lizenzdatei aktualisiert wird, werden nur die Lizenzen darin, die sich noch nicht im Pool befinden, in den Pool geladen. Um eine Lizenz zu aktivieren, zu deaktivieren oder zu löschen, wählen Sie sie aus und klicken Sie auf die Schaltfläche **Activate**, **Deactivate** bzw. **Delete**.

Aktivieren der gewünschten Lizenz(en)
Damit Sie einem Altova-Produkt eine Lizenz zuweisen können, muss sie aktiv sein. Wenn sie inaktiv ist, wählen Sie sie aus und klicken Sie auf **Activate**.

 © 2014 Altova GmbH

Nächste Schritte
Nachdem Sie die Lizenzdatei in den LicenseServer geladen und sichergestellt haben, dass sie aktiv ist, gehen Sie folgendermaßen vor:

1. Registrieren Sie das Altova Server-Produkt (FlowForce Server, MapForce Server, StyleVision Server) auf dem LicenseServer. Wenn Sie dies bereits vor dem Laden der Lizenz getan haben, können Sie nun damit beginnen, Lizenzen zuzuweisen.
2. Weisen Sie Ihrem beim LicenseServer registrierten Altova-Produkt eine Lizenz zu.

10.6.6 Registrieren von Produkten

Bevor Sie einem Altova Server-Produkt eine Lizenz zuweisen können, müssen Sie die Produktinstallation am LicenseServer registrieren. Die Registrierung wird vom Altova Server-Produkt aus durchgeführt und das Verfahren ist unterschiedlich, je nachdem, ob das Server-Produkt eine Web-Benutzeroberfläche hat oder ob es über die Befehlszeile ausgeführt wird. Für die Registrierung benötigen Sie den Server-Namen oder die IP-Adresse des Rechners, auf dem LicenseServer installiert ist.

In diesem Abschnitt wird beschrieben, wie Sie die verschiedenen Altova Server-Produkte registrieren:

- Registrieren von FlowForce Server
- Registrieren von MapForce Server
- Registrieren von StyleVision Server
- Registrieren von RaptorXML(+XBRL) Server
- Registrieren von MobileTogether Server

Registrieren von FlowForce Server

In diesem Abschnitt werden folgende Schritte beschrieben:

- Methoden, um FlowForce Server zu registrieren
- Aufrufen der FlowForce Server Setup-Seite (Windows)
- Aufrufen der FlowForce Server Setup-Seite (Linux)
- Registrieren von FlowForce Server über die Setup-Seite
- Registrieren von FlowForce Server über das FlowForce CLI (Windows)
- Registrieren von FlowForce Server über das FlowForce CLI (Linux)
- Nächste Schritte

Methoden, um FlowForce Server zu registrieren
FlowForce Server kann auf folgende Arten auf dem LicenseServer registriert werden.

- über die FlowForce Server Setup-Seite
- über das FlowForce CLI (Windows)
- über das FlowForce CLI (Linux)

Aufrufen der FlowForce Server Setup-Seite (Windows)
Die FlowForce Server Setup-Seite kann auf zwei Arten aufgerufen werden:

- Über das Menü **Start**:
  Start | *Altova FlowForce Server 2015* | *FlowForce Server Setup-Seite*
- Über Altova ServiceController: Klicken sie in der Task-Leiste auf das ServiceController-Symbol und wählen Sie im angezeigten Menü den Eintrag *Altova FlowForce Web* | *Setup*.

Daraufhin wird die FlowForce Server Setup-Seite (*Abbildung oben*) aufgerufen.

 © 2014 Altova GmbH

Aufrufen der FlowForce Server Setup-Seite (Linux)
Nachdem Sie FlowForce Server unter Linux installiert haben (eine Beschreibung dazu finden Sie in der Benutzerdokumentation zu FlowForce Server), starten Sie FlowForce Web Server mit dem folgenden Befehl als Dienst:

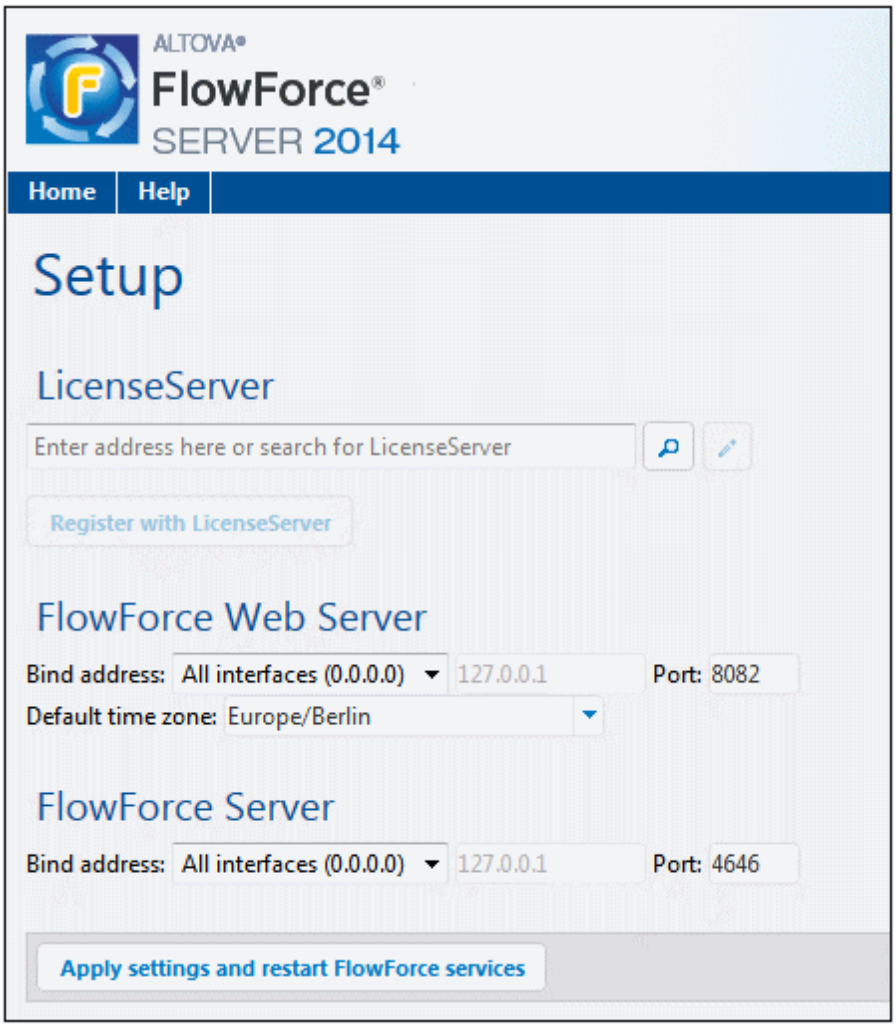
```
sudo /etc/init.d/flowforcewebserver start
```

Im Terminal-Fenster wird eine Meldung, die die URL der FlowForce Server Setup-Seite enthält, angezeigt:

FlowForceWeb running on `http://127.0.1.1:3459/setup?key=52239315203`

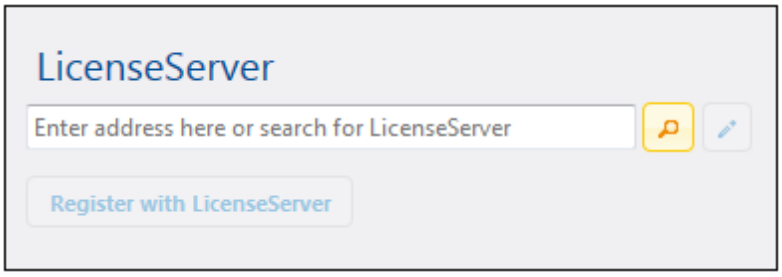
Geben Sie die URl in die Adressleiste eines Browsers ein und drücken Sie die Eingabetaste, um die FlowForce Server Setup-Seite aufzurufen (*Abbildung unten*).

Registrieren von FlowForce Server über die Setup-Seite
Auf der Setup-Seite (*Abbildung unten*) (Anleitung zum Aufruf siehe oben) wird Im Feld LicenseServer angegeben, bei welchem Altova LicenseServer der FlowForce Server registriert werden soll.

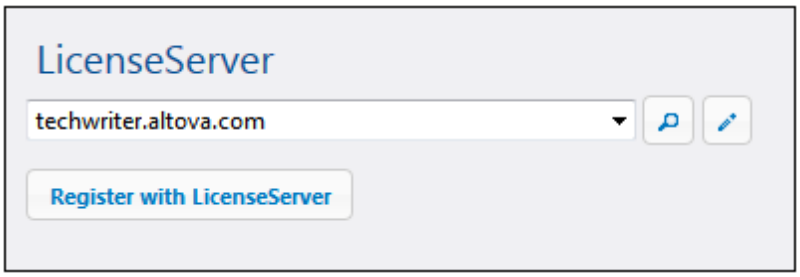


Der gewünschte LicenseServer kann auf zwei verschiedene Arten definiert werden.

- Sie können nach Altova LicenseServern suchen, die derzeit auf dem Netzwerk verfügbar sind, d.h. die gerade ausgeführt werden. Klicken Sie dazu auf die Schaltfläche **Search for Altova LicenseServers** (*in der Abbildung unten gelb markiert*).



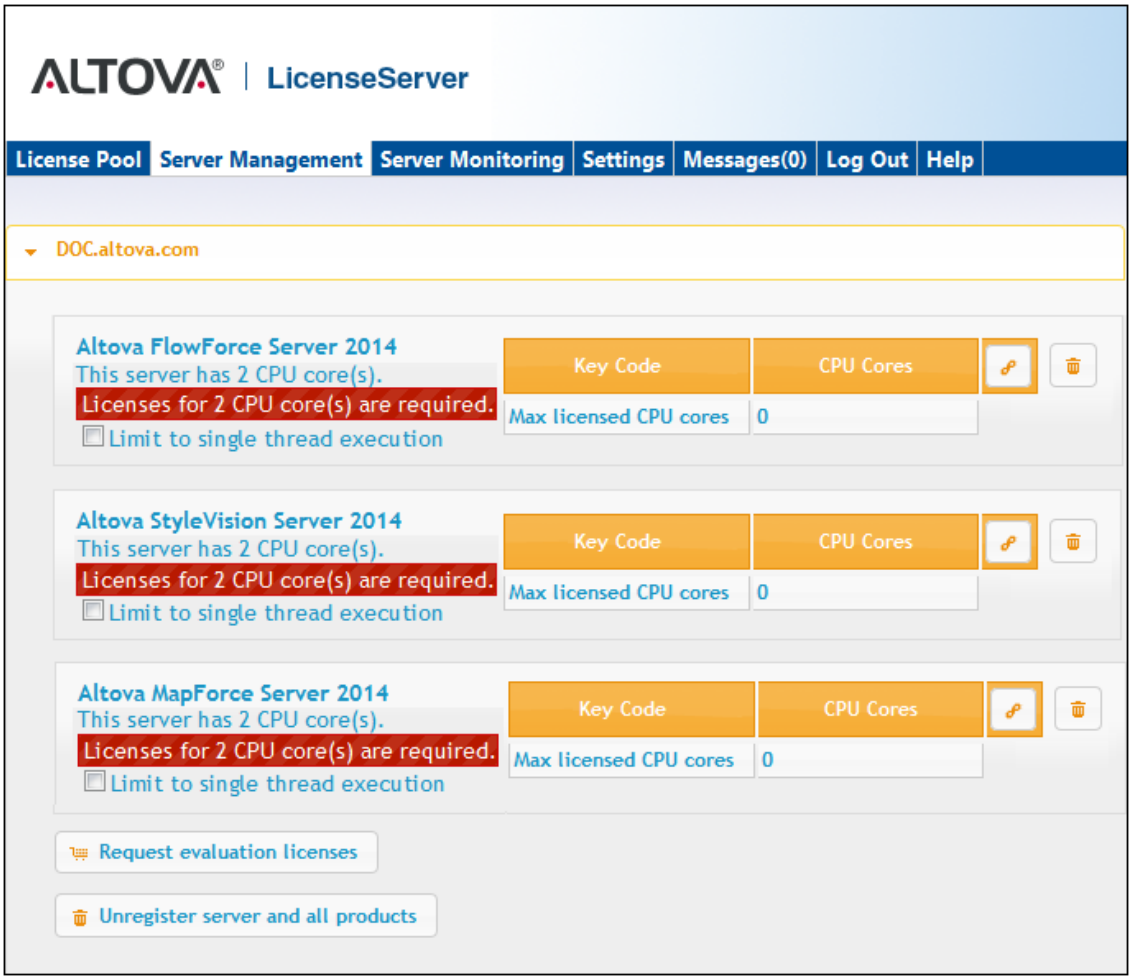
Daraufhin wird eine Liste aller auf dem Netzwerk verfügbaren Altova LicenseServer zurückgegeben. Ein LicenseServer ist ausgewählt (*Abbildung unten*) und die anderen stehen in der Dropdown-Liste der Auswahlliste zur Verfügung. Wählen Sie den LicenseServer aus, auf dem Ihre FlowForce Lizenz gespeichert ist.



- Alternativ dazu können Sie die Adresse des LicenseServers in das Feld LicenseServer eingeben. Wenn die aktuell ausgeführten LicenseServer auch in Form einer Dropdown-Liste zur Verfügung stehen, klicken Sie auf die Schaltfläche **Manually Enter Address,** um eine Adresse in das Feld "LicenseServer" eingeben zu können.

Nachdem Sie den LicenseServer definiert haben, klicken Sie auf **Register with LicenseServer**. Die Altova Server-Applikation wird beim angegebenen LicenseServer registriert und die Konfigurationsseite dieses LicenseServers wird in einem Browser geöffnet, wobei das Register "Server Management" (*Abbildung unten*) aktiv ist.

Anmerkung: Sie müssen unter Umständen die Anzeige von Popup-Fenstern gestatten, damit die LicenseServer Konfigurationsseite angezeigt wird.



In der Abbildung oben wurden drei Altova Produkte beim Altova LicenseServer unter `DOC.altova.com` registriert. Wie man eine Lizenz zuweist, ist im nächsten Abschnitt Zuweisen von Lizenzen zu registrierten Produkten beschrieben.

Registrieren von FlowForce Server über das FlowForce CLI (Windows)
Auf Windows-Rechnern, kann FlowForce Server auch über die Befehlszeile (CLI) auf einem LicenseServer in Ihrem Netzwerk registriert werden. Verwenden Sie dazu den Befehl `licenseserver`:

```
FlowForceServer licenseserver Server-Or-IP-Address
```

Wenn LicenseServer z.B. unter `http://localhost:8088` ausgeführt wird, so registrieren Sie FlowForce Server mit dem folgenden Befehl:

```
FlowForceServer licenseserver localhost
```

Wenn FlowForce Server zusammen im Paket mit anderen Altova Server-Produkten installiert wurde, werden bei der Registrierung von FlowForce Server automatisch auch die Altova Server-Produkte registriert. Nachdem Sie FlowForce Server erfolgreich registriert haben, können Sie zu LicenseServer wechseln und FlowForce Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Registrieren von FlowForce Server über das FlowForce CLI (Linux)
Auf Linux-Rechnern kann FlowForce Server mit dem Befehl `licenseserver` des FlowForce Server CLI am LicenseServer registriert werden. Beachten Sie, dass FlowForce Server mit root-Rechten gestartet werden muss.

```
sudo /opt/Altova/FlowForceServer2015/bin/flowforceserver licenseserver
localhost
```

In der Befehlszeile oben ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie, dass der Pfad der ausführbaren Datei von FlowForce Server der folgende ist:

```
/opt/Altova/MapForceServer2015/bin
```

Nachdem Sie FlowForce Server erfolgreich registriert haben, können Sie zu LicenseServer wechseln und FlowForce Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Nächste Schritte
Nachdem Sie Ihr Altova-Produkt am LicenseServer registriert haben, gehen Sie folgendermaßen vor:

 © 2014 Altova GmbH

1. Wenn Sie die Lizenzdatei(en) mit der/den Lizenzen noch nicht am LicenseServer geladen haben (siehe vorheriger Abschnitt Laden von Lizenzen in LicenseServer), laden Sie sie jetzt und überprüfen Sie, ob die gewünschte Lizenz aktiv ist. Falls Sie das bereits getan haben, fahren Sie mit dem nächsten Schritt, der Zuweisung der Lizenzen fort.
2. Weisen Sie Ihrem beim LicenseServer registrierten Altova-Produkt eine Lizenz zu.

Registrieren von MapForce Server

In diesem Abschnitt werden folgende Schritte beschrieben:

- Registrieren von MapForce Server über FlowForce Server (Windows)
- Registrieren des eigenständigen MapForce Server-Produkts (Windows)
- Registrieren von MapForce Server (Linux)
- Nächste Schritte

MapForce Server kann als Teil des FlowForce Server Pakets oder als eigenständiges Server-Produkt installiert werden. In beiden Fällen muss es am Altova LicenseServer registriert werden. Erst dann können Sie dem Produkt über den LicenseServer eine Lizenz zuweisen. Wenn MapForce Server auf Windows-Systemen im Rahmen des FlowForce Server-Pakets installiert wurde, wird das Produkt automatisch registriert, wenn FlowForce registriert wird. Auf Linux-Systemen wird MapForce Server nur dann automatisch bei der Registrierung von FlowForce Server registriert, wenn es nach FlowForce Server installiert wurde.

Registrieren von MapForce Server über FlowForce Server (Windows)
MapForce Server ist zusammen in einem Paket mit FlowForce Server enthalten. Wenn daher FlowForce Server auf einem Altova LicenseServer in Ihrem Netzwerk installiert wird, wird auch MapForce Server automatisch auf dem LicenseServer registriert. Eine Beschreibung zum Registrieren von FlowForce Server finden Sie in der Dokumentation zu Altova FlowForce Server und im Abschnitt Registrieren von FlowForce Server am LicenseServer.

Nachdem Sie das Produkt registriert haben, können Sie LicenseServer aufrufen und MapForce Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Registrieren des eigenständigen MapForce Server-Produkts (Windows)
Wenn Sie MapForce Server als eigenständiges Paket installiert haben, müssen Sie das Produkt auf einem Altova LicenseServer in Ihrem Netzwerk registrieren und ihm über den Altova LicenseServer eine Lizenz zuweisen. Sie können MapForce Server über seine Befehlszeilenschnittstelle (CLI) mit Hilfe des Befehls `licenseserver` registrieren.

```
MapForceServer licenseserver Server-Or-IP-Address
```

Wenn LicenseServer z.B. unter `http://localhost:8088` ausgeführt wird, so registrieren Sie MapForce Server mit dem folgenden Befehl:

```
MapForceServer licenseserver localhost
```

Nachdem Sie MapForce Server erfolgreich registriert haben, können Sie LicenseServer aufrufen und StyleVision Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Registrieren von MapForce Server (Linux)
Auf Linux-Rechnern kann MapForce Server mit dem Befehl `licenseserver` des MapForce Server CLI am LicenseServer registriert werden. Beachten Sie, dass MapForce Server mit root-Rechten gestartet werden muss.

```
sudo /opt/Altova/MapForceServer2015/bin/mapforceserver licenseserver
localhost
```

In der Befehlszeile oben ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie, dass der Pfad der ausführbaren Datei von MapForce Server der folgende ist:

```
/opt/Altova/MapForceServer2015/bin
```

Nachdem Sie MapForce Server erfolgreich registriert haben, können Sie zu LicenseServer wechseln und MapForce Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Nächste Schritte
Nachdem Sie Ihr Altova-Produkt am LicenseServer registriert haben, gehen Sie folgendermaßen vor:

1. Wenn Sie die Lizenzdatei(en) mit der/den Lizenzen noch nicht am LicenseServer geladen haben (siehe vorheriger Abschnitt Laden von Lizenzen in LicenseServer), laden Sie sie jetzt und überprüfen Sie, ob die gewünschte Lizenz aktiv ist. Falls Sie das bereits getan haben, fahren Sie mit dem nächsten Schritt, der Zuweisung der Lizenzen fort.
2. Weisen Sie Ihrem beim LicenseServer registrierten Altova-Produkt eine Lizenz zu.

Registrieren von StyleVision Server

In diesem Abschnitt werden folgende Schritte beschrieben:

- Registrieren von StyleVision Server über FlowForce (Windows)
- Registrieren des eigenständigen StyleVision Server-Produkts (Windows)
- Registrieren von StyleVision Server (Linux)
- Nächste Schritte

 © 2014 Altova GmbH

StyleVision Server kann als Teil des FlowForce Server Pakets oder als eigenständiges Server-Produkt installiert werden. In beiden Fällen muss es am Altova LicenseServer registriert werden. Erst dann können Sie dem Produkt über den LicenseServer eine Lizenz zuweisen. Wenn StyleVision Server auf Windows-Systemen im Rahmen des FlowForce Server-Pakets installiert wurde, wird das Produkt automatisch registriert, wenn FlowForce registriert wird. Auf Linux-Systemen wird StyleVision Server nur dann automatisch bei der Registrierung von FlowForce Server registriert, wenn es nach FlowForce Server installiert wurde.

Registrieren von StyleVision Server über FlowForce (Windows)
StyleVision Server ist zusammen in einem Paket mit FlowForce Server enthalten. Wenn daher FlowForce Server auf einem Altova LicenseServer in Ihrem Netzwerk installiert wird, wird auch StyleVision Server automatisch auf dem LicenseServer registriert. Eine Beschreibung zum Registrieren von FlowForce Server finden Sie in der Dokumentation zu Altova FlowForce Server und im Abschnitt Registrieren von FlowForce Server am LicenseServer.

Nachdem Sie das Produkt registriert haben, können Sie LicenseServer aufrufen und StyleVision Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Registrieren des eigenständigen StyleVision Server-Produkts (Windows)
Wenn Sie StyleVision Server als eigenständiges Paket unter Windows installiert haben, müssen Sie das Produkt auf einem Altova LicenseServer in Ihrem Netzwerk registrieren und ihm über den Altova LicenseServer eine Lizenz zuweisen. Sie können StyleVision Server über seine Befehlszeilenschnittstelle (CLI) mit Hilfe des Befehls `licenseserver` registrieren.

```
StyleVisionServer licenseserver [options] Server-Or-IP-Address
```

Wenn LicenseServer z.B. unter `http://localhost:8088` ausgeführt wird, so registrieren Sie StyleVision Server mit dem folgenden Befehl:

```
StyleVisionServer licenseserver localhost
```

Nachdem Sie StyleVision Server erfolgreich registriert haben, können Sie LicenseServer aufrufen und StyleVision Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Registrieren von StyleVision Server (Linux)
Auf Linux-Rechnern kann StyleVision Server mit dem Befehl `licenseserver` des StyleVision Server CLI am LicenseServer registriert werden. Beachten Sie, dass StyleVision Server mit root-Rechten gestartet werden muss.

```
sudo /opt/Altova/StyleVisionServer2015/bin/stylevisionserver licenseserver
localhost
```

In der Befehlszeile oben ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie, dass der Pfad der ausführbaren Datei von StyleVision Server der folgende ist:

```
/opt/Altova/StyleVisionServer2015/bin
```

Nachdem Sie StyleVision Server erfolgreich registriert haben, können Sie zu LicenseServer wechseln und StyleVision Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Nächste Schritte
Nachdem Sie Ihr Altova-Produkt am LicenseServer registriert haben, gehen Sie folgendermaßen vor:

1. Wenn Sie die Lizenzdatei(en) mit der/den Lizenzen noch nicht am LicenseServer geladen haben (siehe vorheriger Abschnitt Laden von Lizenzen in LicenseServer), laden Sie sie jetzt und überprüfen Sie, ob die gewünschte Lizenz aktiv ist. Falls Sie das bereits getan haben, fahren Sie mit dem nächsten Schritt, der Zuweisung der Lizenzen fort.
2. Weisen Sie Ihrem beim LicenseServer registrierten Altova-Produkt eine Lizenz zu.

Registrieren von RaptorXML(+XBRL) Server

In diesem Abschnitt werden folgende Schritte beschrieben:

- Registrieren von RaptorXML(+XBRL) Server (Windows)
- Registrieren von RaptorXML(+XBRL) Server (Linux)
- Nächste Schritte

RaptorXML(+XBRL) Server muss auf dem Server-Rechner oder in dem Netzwerk installiert werden, mit dem LicenseServer verbunden ist, und anschließend als Dienst gestartet werden. Anschließend muss das Produkt auf dem LicenseServer registriert werden. Erst danach kann dem Produkt über LicenseServer eine Lizenz zugewiesen werden. In diesem Abschnitt wird beschrieben, wie Sie RaptorXML(+XBRL) Server auf dem LicenseServer registrieren.

Registrieren von RaptorXML(+XBRL) Server (Windows)
Sie können RaptorXML(+XBRL) Server mit dem Befehl `licenseserver` über seine Befehlszeilenschnittstelle (CLI) registrieren:

RaptorXML Server:	**RaptorXML licenseserver *Server-Or-IP-Address***
RaptorXML+XBRL Server:	**RaptorXMLXBRL licenseserver *Server-Or-IP-Address***

 © 2014 Altova GmbH

Wenn LicenseServer z.B. unter `http://localhost:8088` ausgeführt wird, so registrieren Sie RaptorXML (+XBRL) Server mit dem folgenden Befehl:

RaptorXML Server:	**`RaptorXML licenseserver localhost`**
RaptorXML+XBRL Server:	**`RaptorXMLXBRL licenseserver localhost`**

Nachdem Sie RaptorXML(+XBRL) Server registriert haben, können Sie LicenseServer aufrufen und RaptorXML(+XBRL) Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Registrieren von RaptorXML(+XBRL) Server (Linux)
Auf Linux-Rechnern kann RaptorXML(+XBRL) Server mit dem Befehl `licenseserver` des RaptorXML(+XBRL) Server CLI am LicenseServer registriert werden. Beachten Sie, dass RaptorXML(+XBRL) Server mit root-Rechten gestartet werden muss.

```
sudo /opt/Altova/RaptorXMLServer2015/bin/raptorxmlserver licenseserver
localhost
sudo /opt/Altova/RaptorXMLXBRLServer2015/bin/raptorxmlxbrlserver licenseserver
localhost
```

In der Befehlszeile oben ist `localhost` der Name des Servers, auf dem LicenseServer installiert ist. Beachten Sie, dass der Pfad der ausführbaren Datei von RaptorXML(+XBRL) Server der folgende ist:

```
/opt/Altova/RaptorXMLServer2015/bin
/opt/Altova/RaptorXMLXBRLServer2015/bin
```

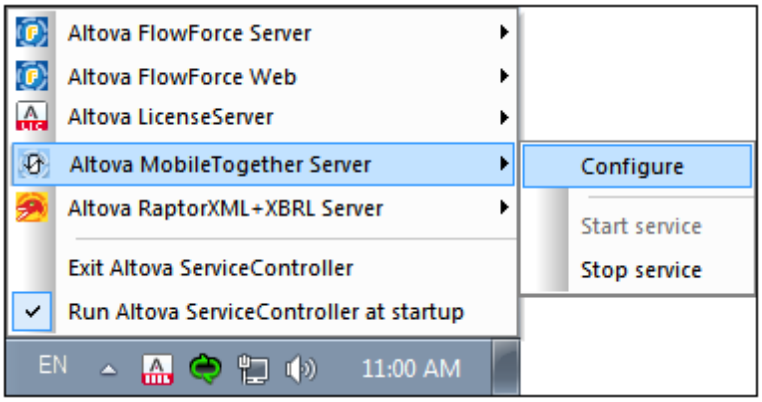
Nachdem Sie RaptorXML(+XBRL) Server erfolgreich registriert haben, können Sie zu LicenseServer wechseln und RaptorXML(+XBRL) Server eine Lizenz zuweisen. Eine Beschreibung dazu finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

Nächste Schritte
Nachdem Sie Ihr Altova-Produkt am LicenseServer registriert haben, gehen Sie folgendermaßen vor:

1. Wenn Sie die Lizenzdatei(en) mit der/den Lizenzen noch nicht am LicenseServer geladen haben (siehe vorheriger Abschnitt Laden von Lizenzen in LicenseServer), laden Sie sie jetzt und überprüfen Sie, ob die gewünschte Lizenz aktiv ist. Falls Sie das bereits getan haben, fahren Sie mit dem nächsten Schritt, der Zuweisung der Lizenzen fort.
2. Weisen Sie Ihrem beim LicenseServer registrierten Altova-Produkt eine Lizenz zu.

Registrieren von MobileTogether Server

Um MobileTogether Server zu starten, klicken Sie in der Taskleiste auf das **ServiceController** Symbol, platzieren Sie den Mauszeiger im daraufhin angezeigten Menü (*siehe Abbildung unten*) über **Altova MobileTogether Server** und wählen Sie aus dem Untermenü den Befehl **Start Service** aus. Wenn MobileTogether Server bereits ausgeführt wird, ist die Option *Start Service* deaktiviert.



Registrieren von MobileTogether Server über:

- das Register "Einstellungen" der MobileTogether Server Web-Schnittstelle: (i) Starten Sie MobileTogether Server über den ServiceController (*siehe vorhergehender Punkt*); (ii) Geben Sie Ihr Passwort ein, um die Setup-Seite aufzurufen; (iii) Wählen Sie den Namen oder die Adresse des LicenseServers aus und klicken Sie auf **Auf LicenseServer registrieren**.

- das CLI (die Befehlszeilenschnittstelle) mit dem Befehl `licenseserver`:
  `MobileTogetherServer licenseserver [options]` ***`ServerName-Or-IP-Address`***
  Wenn z.B. `localhost` der Name des Servers ist, auf dem LicenseServer installiert ist:
  `MobileTogetherServer licenseserver localhost`

Nachdem Sie MobileTogether Server erfolgreich registriert haben, rufen Sie das Register "Server Management" der Konfigurationsseite von LicenseServer auf, um MobileTogether Server eine Lizenz zuzuweisen.

 © 2014 Altova GmbH

10.6.7 Zuweisen von Lizenzen zu registrierten Produkten

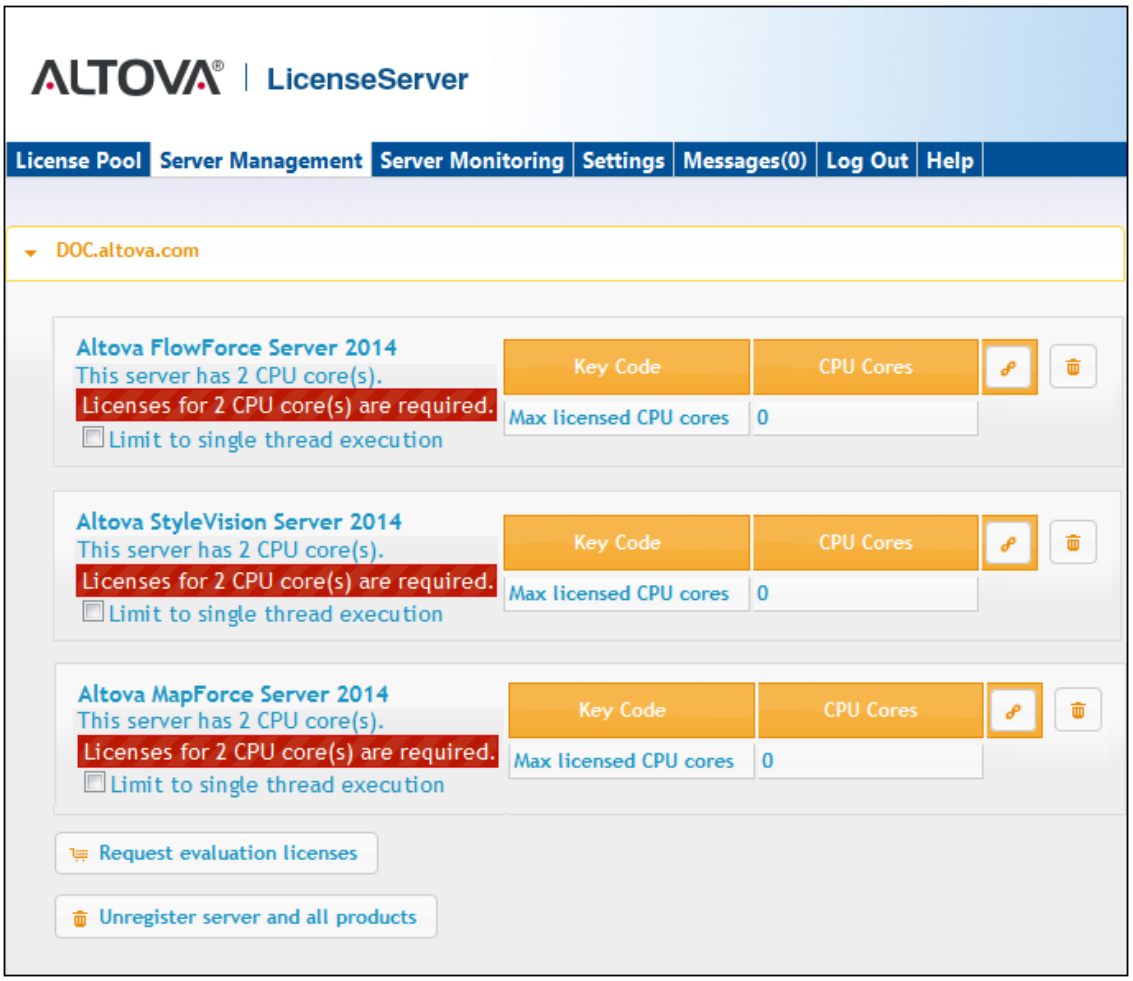
In diesem Abschnitt werden folgende Schritte beschrieben:

- Vor dem Zuweisen einer Lizenz
- Das Register "Server Management"
- Symbole auf dem Register "Server Management"
- Anmerkung zu Kernen und Lizenzen
- Zuweisen eine Lizenz
- Aufheben der Registrierung von Produkten am LicenseServer

Vor dem Zuweisen einer Lizenz
Bevor Sie einem Altova-Produkt eine Lizenz zuweisen können, stellen Sie sicher, dass:

- die entsprechende Lizenz in den Lizenzpool von LicenseServer geladen wurde und dass die Lizenz aktiv ist.
- Ihr Altova-Produkt beim LicenseServer registriert wurde.

Das Register "Server Management"
Die Lizenzzuweisung erfolgt auf dem Register "Server Management" der LicenseServer-Konfigurationsseite (*Abbildung unten*). In der Abbildung sehen Sie, dass drei Altova-Produkte beim LicenseServer registriert sind. (Da MapForce Server und StyleVision Server mit FlowForce Server gebündelt sind, werden bei der Registrierung von FlowForce Server beim LicenseServer automatisch auch MapForce Server und StyleVision Server registriert. Wenn FlowForce Server registriert wurde, müssen die beiden anderen Produkte nicht mehr separat registriert werden.)



Beachten Sie zum Register "Server Management" die folgenden Punkte:

- Jedes Produkt wird unter dem Namen seines Client-Rechners aufgelistet. In der Abbildung oben ist ein Client-Rechner namens `Doc.altova.com` aufgelistet. Unter diesem Namen (`Doc.altova.com`) sind drei Altova-Produkte auf dem LicenseServer registriert. Wenn ein auf einem anderen Client-Rechner installiertes Altova-Produkt bei diesem LicenseServer registriert ist, so wird dieser Client-Rechner mit seinen registrierten Produkten ebenfalls auf dem Register "Server Management" aufgelistet.
- Jedes registrierte Altova-Produkt auf einem Client-Rechner hat seinen eigenen *Keycode* Eintrag, d.h. den Keycode aus einer Lizenz. Der Keycode eines registrierten Produkts wird durch klicken auf die Schaltfläche **Edit assigned Licenses** (siehe Symbolliste unten) und Auswahl der erforderlichen Lizenzen aus den im Lizenzpool für das Produkt (z.B. FlowForce Server) verfügbaren Lizenzen zugewiesen. Eine ausführlichere Anleitung dazu finden Sie weiter unten.
- Außerdem gibt es zu jedem Produkt eine Zeile, in der angegeben wird, wie viele CPU-Kerne für dieses Produkt auf diesem Client lizenziert werden müssen. Wenn die Anzahl der lizenzierten Kerne kleiner als die erforderliche Anzahl ist, wird diese Information rot markiert (*siehe Abbildung oben*). (Die Anzahl der zu lizenzierenden CPU-Kerne entspricht der Anzahl der CPU-Kerne auf diesem Client. Diese Zahl wird von LicenseServer vom Client-Rechner abgerufen.)
- Wenn auf demselben Rechner **mehrere Versionen** eines einzigen Produkts (z.B. StyleVision Server 2013 und StyleVision Server 2014) installiert wurden und wenn jede

dieser Installationen auf einem einzigen LicenseServer registriert wurde, so werden diese Mehrfachregistrierungen auf dem Register "Server Management" zu einer einzigen Registrierung zusammengefasst und als nur eine Registrierung angezeigt. Wenn dieser einzigen Registrierung eine Lizenz zugewiesen wird, werden alle Installationen zu dieser Registrierung lizenziert. Es können jedoch auf dem Client-Rechner nur mehrere Instanzen einer einzigen Installation gleichzeitig ausgeführt werden. So können etwa mehrere Instanzen von StyleVision Server 2013 oder mehrere Instanzen von StyleVision Server 2014 gleichzeitig ausgeführt werden, nicht aber eine Instanz von StyleVision Server 2013 und eine Instanz von StyleVision Server 2014. Beachten Sie, dass neu installierte Versionen zuerst registriert werden müssen, bevor sie ausgeführt werden können.

- Neue Versionen von Altova Server-Produkten können nur mit der Version von LicenseServer lizenziert werden, die zum Zeitpunkt der Produkt-Release die neueste ist, jedoch funktionieren ältere Versionen von Altova Server-Produkten mit neueren Versionen des LicenseServer. Wenn Sie daher eine neue Version eines Altova-Produkts installieren und Ihre aktuelle LicenseServer-Version nicht die neueste ist, deinstallieren Sie die ältere Version von LicenseServer und installieren Sie die neueste Version. Alle Registrierungs- und Lizenzierungsdaten aus Ihrer alten Version von LicenseServer werden bei der Deinstallation in einer Datenbank auf dem Server gespeichert und automatisch in die neuere Version importiert.) (Welche LicenseServer-Version für eine bestimmte Version eines Server-Produkts benötigt wird, wird während der Installation des Server-Produkts angezeigt. Sie können diese Version von LicenseServer zusammen mit dem Server-Produkt installieren. Die Nummer der aktuell installierten LicenseServer-Version wird am unteren Rand der LicenseServer-Konfigurationsseite angezeigt.)

Symbole auf dem Register "Server Management"

Edit Assigned Licenses. (Zugewiesene Lizenzen bearbeiten) Steht für jedes Produkt zur Verfügung. Ruft das Dialogfeld "Manage Licenses" auf, in dem dem Produkt neue Lizenzen zugewiesen und bereits zugewiesene bearbeitet werden können.

Show Licenses. (Lizenzen anzeigen) Wird zu jeder Lizenz angezeigt. Damit wechseln Sie zum Register "License Pool", wo die ausgewählte Lizenz markiert wird, damit Sie die Lizenzdetails lesen können.

Unregister This Product. (Registrierung für dieses Produkt aufheben) Steht für jedes Produkt zur Verfügung. Die Registrierung des ausgewählten Produkts (auf dem ausgewählten Client-Rechner) wird beim LicenseServer aufgehoben.

Anmerkung zu Prozessorkernen und Lizenzen
Die Lizenzierung von Altova Server-Produkten basiert *mit Ausnahme von MobileTogether Server*** auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor hat vier Kerne, ein Hexa-Core-Prozessor hat sechs Kerne, usw. Die Anzahl der für ein Produkt auf einem bestimmten Server-Rechner lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Server verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt.

Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie eine Lizenz für mindestens acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die

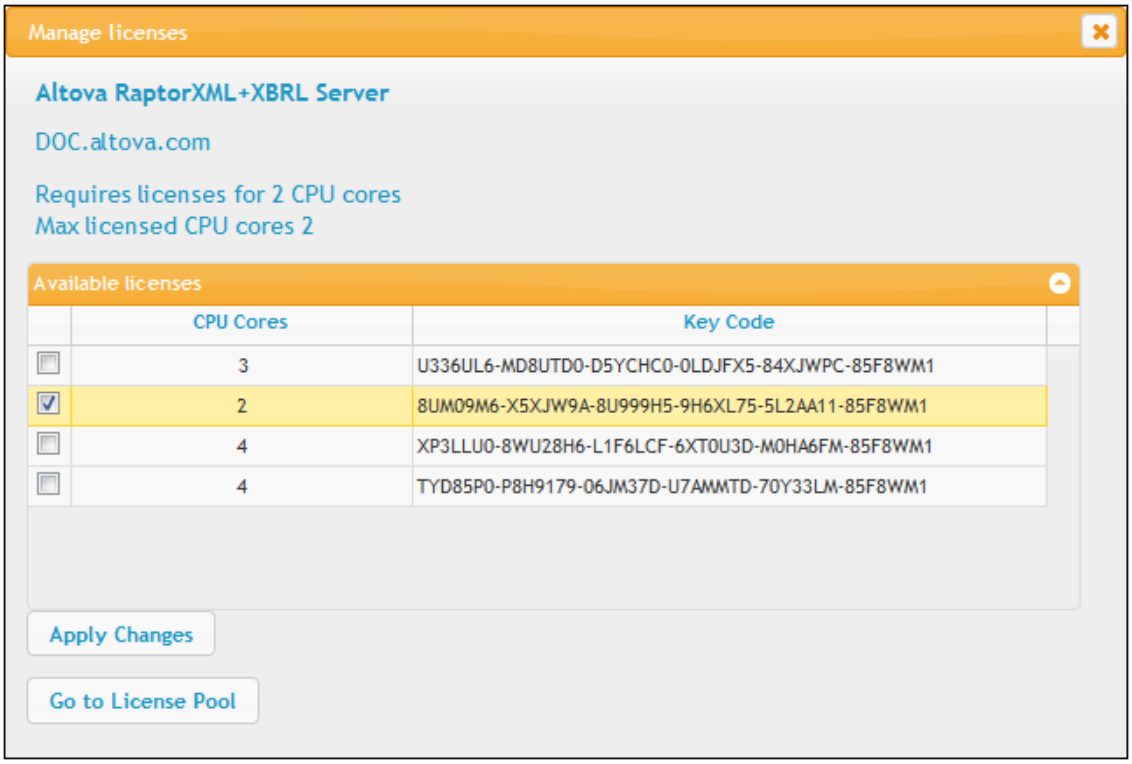
entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen Octa-Core-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit eine großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Lizenz kann immer nur für einen Client gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität noch nicht ausgeschöpft ist. Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

*** MobileTogether Server-Lizenzen werden auf Basis der Anzahl der Benutzer, d.h. auf Basis der Clients, die auf MobileTogether Server zugreifen, zugewiesen.

Zuweisen einer Lizenz
Um einem registrierten Produkt eine Lizenz zuzuweisen, klicken Sie auf die Schaltfläche **Edit Assigned Licenses** dieses Produkts. Daraufhin wird das Dialogfeld "Manage Licenses" (*Abbildung unten*) aufgerufen.



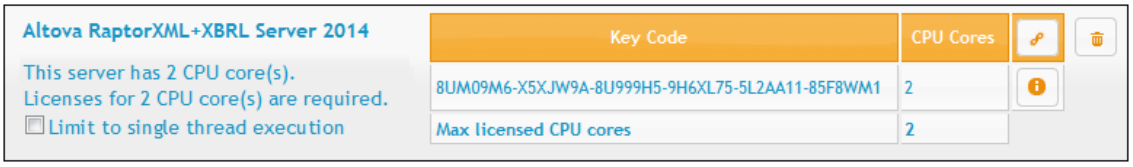
Beachten Sie zu den im Dialogfeld "Manage Licenses" angezeigten Lizenzen die folgenden Punkte:

 © 2014 Altova GmbH

- Das zu lizenzierende Produkt wird links oben im Dialogfeld angeführt. In der Abbildung oben ist dies Altova FlowForce Server 2013.
- Im Dialogfeld werden alle derzeit aktiven Lizenzen für dieses Produkt im Lizenzpool angezeigt. In unserer Abbildung befinden sich vier derzeit aktive FlowForce Server Lizenzen im Lizenzpool.LicenseServer erkennt automatisch anhand jeder Lizenz im Pool, für welches Altova-Produkt sie ausgestellt wurde.
- Die Lizenzen in der Abbildung oben gelten für 3 CPU-Kerne, 2 CPU-Kerne, 4 CPU-Kerne bzw. 4 CPU-Kerne.
- Sie müssen die Anzahl der Prozessorkerne auf dem Server, auf dem das Altova Server-Produkt installiert wurde, kennen. Wenn der Rechner einen Dual-Core-Prozessor hat, benötigen Sie eine Lizenz für 2 Kerne (Anzahl der CPU-Kerne). Dabei könnte es sich in der Liste in der Abbildung oben z.B. um die zweite Lizenz handeln. Sie können Lizenzen auch kombinieren. Wenn der Rechner z.B. einen Octa-Core-Prozessor hat (acht Kerne), können Sie zwei Lizenzen für 4 Kerne kombinieren, z.B. die dritte und vierte Lizenz in der Liste in der Abbildung oben.
- Im Dialogfeld "Manage Licenses" werden nur aktive Lizenzen für dieses Produkt aufgelistet. Lizenzen für andere Altova-Produkte werden nicht aufgelistet.
- Lizenzen, die bereits z.B. einer anderen Installation dieses Produkts im Netzwerk zugewiesen wurden, sind mit einem Häkchen versehen. Nur freie Lizenzen, die mit keinem Häkchen versehen sind, können ausgewählt werden.
- Mit CPU cores wird angegeben, für wie viele CPU-Kerne eine Lizenz gültig ist.
- Wenn Sie am Lizenzpool Änderungen vornehmen möchten, z.B. um eine Lizenz zu laden, zu aktivieren, zu deaktivieren oder zu löschen, klicken Sie auf die Schaltfläche **Go to License Pool**.

Wählen Sie die gewünschte Lizenz aus. Die Lizenz wird mit einem Häkchen versehen. Die Gesamtanzahl der CPU-Kerne, für die diese Lizenz für dieses Produkt auf diesem Client gültig ist, wird im linken oberen Bereich des Dialogfelds als *Max licensed CPU cores* aufgelistet (*siehe Abbildung oben*). Sie können mehr Lizenzen auswählen, wenn das installierte Produkt auf diesem Client für mehr CPU-Kerne lizenziert werden soll. In diesem Fall wird unter *Max licensed CPU cores* die Summe aller CPU-Kerne aller ausgewählten Lizenzen angegeben.

Klicken Sie nach Auswahl der Lizenz(en) auf **Apply Changes**. Daraufhin wird/werden die Lizenz(en) diesem Produkt zugewiesen und auf dem Register "Server Management" angezeigt (*siehe Abbildung unten*). In der Abbildung unten sehen Sie, dass (dem Client-Rechner `Doc.altova.com`) eine 2-Kern-Lizenz für Altova FlowForce Server zugewiesen wurde.



Aufheben der Registrierung für Produkte
Jedes bei LicenseServer registrierte Altova-Produkt wird auf dem Register "Server Management" unter dem Namen seines Client-Rechners aufgelistet. Rechts daneben befindet sich ein **Unregister** Symbol. Klicken Sie darauf, um die Registrierung für das Produkt aufzuheben. Wenn dem Produkt eine Lizenz zugewiesen wurde, wird die Zuweisung beendet, sobald die

Registrierung für das Produkt aufgehoben wird. Um die Registrierung für alle Produkte aufzuheben, klicken Sie am unteren Rand des Registers "Server Management" auf die Schaltfläche **Unregister Server and All Products** (*siehe erste Abbildung in diesem Abschnitt)*.

Um ein Produkt erneut zu registrieren, gehen Sie zur Vorkonfigurationsseite des Produkts.

© 2014 Altova GmbH

10.7 Referenz zur Konfigurationsseite

Die LicenseServer-Konfigurationsseite ist die Administratorschnittstelle zu LicenseServer (Web-Benutzeroberfläche). Sie dient zum Verwalten von LicenseServer und zum Lizenzieren von Altova-Produkten (FlowForce Server, MapForce Server, StyleVision Server, RaptorXML(+XBRL) Server), die auf dem LicenseServer registriert wurden. Die LicenseServer-Konfigurationsseite wird über einen Webbrowser aufgerufen. Eine Beschreibung, wie Sie die Konfigurationsseite aufrufen, finden Sie in den Abschnitten Öffnen der LicenseServer-Konfigurationsseite (Windows) und Öffnen der LicenseServer-Konfigurationsseite (Linux).

Dieser Abschnitt ist eine Benutzerreferenz zur Konfigurationsseite und ist nach den Registern der Konfigurationsseite gegliedert:

- License Pool
- Server Management
- Server Monitoring
- Settings
- Messages, Log Out

Eine schrittweise Anleitung zum Zuweisen von Lizenzen über LicenseServer finden Sie im Abschnitt Zuweisen von Lizenzen.

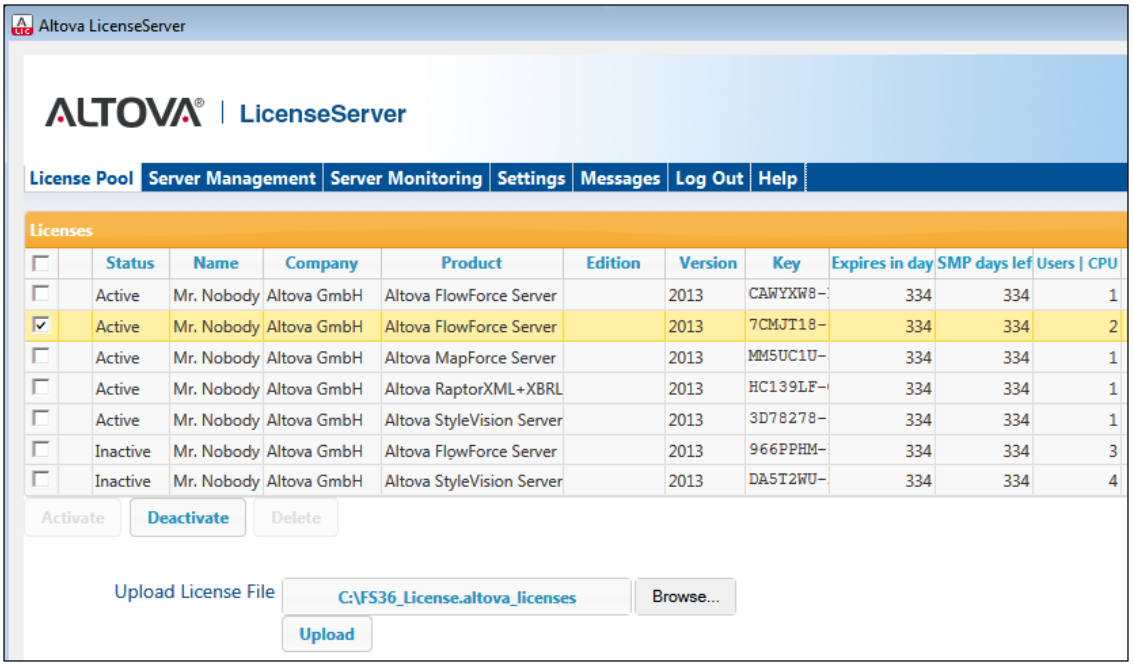
10.7.1 License Pool

In diesem Abschnitt werden folgende Schritte beschrieben:

- Laden einer Lizenz
- Lizenzstatus
- Aktivieren, Deaktivieren und Löschen einer Lizenz
- Symbole auf dem Register "License Pool"
- Lizenzinformationen
- Anmerkung zu Prozessorkernen und Lizenzen

Auf dem Register **License Pool** werden alle derzeit auf dem LicenseServer gespeicherten Lizenzen angezeigt (*siehe Abbildung unten*). Wenn eine Lizenzdatei über die Schaltfläche **Upload** auf dieser Seite in den LicenseServer geladen wird, werden alle in der Lizenzdatei enthaltenen Lizenzen in den Lizenzpool auf dem Server gestellt und auf der Seite "License Pool" angezeigt.

Auf der Seite "License Pool" werden Informationen über alle Lizenzen, die sich derzeit auf dem LicenseServer befinden, angezeigt, sodass Sie einen Überblick über alle Altova-Produktlizenzen haben. Sie können ausgewählte Lizenzen auf dieser Seite auch aktivieren, deaktivieren und löschen.


Altova LicenseServer
ALTOVA | LicenseServer
License Pool
Server Management
Server Monitoring
Settings
Messages
Log Out
Help
Licenses
Status Name Company Product Edition Version Key Expires in day SMP days lef Users | CPU
Active Mr. Nobody Altova GmbH Altova FlowForce Server 2013 CAWYXW8- 334 334 1
Active Mr. Nobody Altova GmbH Altova FlowForce Server 2013 7CMJT18- 334 334 2
Active Mr. Nobody Altova GmbH Altova MapForce Server 2013 MM5UC1U- 334 334 1
Active Mr. Nobody Altova GmbH Altova RaptorXML+XBRL 2013 HC139LF- 334 334 1
Active Mr. Nobody Altova GmbH Altova StyleVision Server 2013 3D78278- 334 334 1
Inactive Mr. Nobody Altova GmbH Altova FlowForce Server 2013 966PPHM- 334 334 3
Inactive Mr. Nobody Altova GmbH Altova StyleVision Server 2013 DA5T2WU- 334 334 4
Activate
Deactivate
Delete
Upload License File
C:\FS36_License.altova_licenses
Browse...
Upload

Laden einer Lizenz
Um eine Lizenzdatei, die Sie von der Altova GmbH für Ihr Altova Server-Produkt erhalten haben,

zu laden, klicken Sie auf die Schaltfläche **Browse,** navigieren Sie zur Lizenzdatei und wählen Sie sie aus. Sobald Sie auf **Upload** klicken, werden alle in der Lizenzdatei enthaltenen Lizenzen in den Lizenzpool platziert und auf der Seite "License Pool" (*Abbildung oben*) angezeigt. Neu hochgeladene Lizenzen werden automatisch aktiviert und ihr Status wird auf *Active* gesetzt.

Lizenzstatus
Es gibt die folgenden Lizenzstatuswerte:

- *Activating:* Nachdem eine Lizenz in den Lizenzpool von LicenseServer hochgeladen wurde, werden die Lizenzdaten dazu zur Überprüfung, Authentifizierung und Aktivierung der geladenen Lizenz an den Master Licensing Server unter `altova.com` weitergeleitet, um die Einhaltung des Altova Lizenzvertrags zu überwachen. Währenddessen - die Dauer kann je nach Verbindung und Netzwerkverkehr zwischen 30 Sekunden und einigen Minuten variieren - wird der Status als *Activating....*angezeigt.

- *Failed Verification:* Falls keine Verbindung zum Master Licensing Server unter `altova.com` hergestellt werden konnte, wird der Status der Lizenz im Pool als *Failed Verification* angezeigt. Überprüfen Sie in diesem Fall Ihre Internetverbindung und die Firewall-Einstellungen, um sicherzustellen, dass LicenseServer eine Verbindung zum Master Licensing Server unter `altova.com` herstellen kann.

- *Active:* Sobald die Lizenz authentifiziert und aktiviert wurde, ändert sich der Status im Pool in *Active.*

- *Inactive:* Wenn eine Lizenz überprüft wurde, aber auf einem anderen LicenseServer im Netzwerk vorhanden ist, wird der Status im Pool als *Inactive* angezeigt. Der Status *Inactive* kann auch zustande kommen, wenn eine Lizenz im Lizenzpool vom Administrator manuell deaktiviert wurde.

- *Blocked:* Eine Lizenz erscheint im Lizenzpool als *Blocked,* wenn es ein Problem mit der Authentifizierung der Lizenz gab und der Master License Server unter `altova.com` die Verwendung der Lizenz nicht gestattet. Dies kann der Fall sein, wenn der Lizenzvertrag nicht eingehalten wurde, zu viele Lizenzen verwendet werden oder andere Lizenzverletzungen festgestellt wurden. Wird eine Lizenz mit dem Status *Blocked* angezeigt, wenden Sie sich bitte mit Ihren Lizenzdaten und anderen relevanten Daten an den Altova Support.

Eine Übersicht über die Statuswerte finden Sie in der folgenden Tabelle:

Status	Bedeutung
Activating...	Nach dem Laden werden die Lizenzdaten zur Überprüfung an `altova.com` gesendet. Aktualisieren Sie den Browser, um den aktualisierten Status zu sehen. Die Überprüfung und Aktivierung kann einige Minuten dauern.
Failed Verification	Es konnte keine Verbindung zu `altova.com` hergestellt werden. Nachdem eine Verbindung hergestellt wurde, starten Sie den Dienst entweder neu oder aktivieren Sie die Lizenz (über die Schaltfläche **Activate**).
Active	Die Überprüfung war erfolgreich, die Lizenz ist aktiv.

Inactive	Die Überprüfung war erfolgreich, aber die Lizenz wird derzeit auf einem anderen LicenseServer im Netzwerk verwendet. Lizenzen können über die Schaltfläche **Deactivate** deaktiviert werden.
Blocked	Die Überprüfung war negativ. Die Lizenz ist ungültig und wurde blockiert. Wenden Sie sich an den Altova Support

Anmerkung: Nachdem die Lizenz zur Überprüfung an `altova.com` gesendet wurde, muss der Browser aktualisiert werden, damit Sie den aktualisierten Status sehen. Die Überprüfung und Aktivierung kann einige Minuten dauern.

Anmerkung: Falls keine Verbindung zu `altova.com` hergestellt werden konnte, wird der Status als *Failed Verification* angezeigt. Nachdem eine Verbindung hergestellt wurde, starten Sie den Dienst entweder neu oder versuchen Sie, die Lizenz (über die Schaltfläche **Activate**) zu aktivieren.

Anmerkung: Wenn eine Lizenz den Status *Inactive* oder *Blocked* erhält, wird eine Meldung mit einer Erklärung dazu zum Meldungs-Log hinzugefügt.

Nur eine aktive Lizenz kann einem Produkt zugewiesen werden. Eine inaktive Lizenz kann aktiviert oder aus dem Lizenz-Pool gelöscht werden. Wenn eine Lizenz aus dem Lizenz-Pool gelöscht wurde, kann sie erneut durch Laden der darin enthaltenen Lizenzdatei in den Pool geladen werden. Wenn eine Lizenzdatei aktualisiert wird, werden nur die Lizenzen darin, die sich noch nicht im Pool befinden, in den Pool geladen. Um eine Lizenz zu aktivieren, zu deaktivieren oder zu löschen, wählen Sie sie aus und klicken Sie auf die Schaltfläche **Activate**, **Deactivate** bzw. **Delete**.

Verbindung zum Master Licensing Server unter altova.com
Altova LicenseServer muss in Verbindung mit dem Master Licensing Server unter `altova.com` stehen, um Lizenzdaten überprüfen und authentifizieren und die Einhaltung der Altova-Lizenzverträge fortlaufend überwachen zu können. Diese Verbindung erfolgt über HTTPS über den Port 443. Wenn der Altova LicenseServer nach dem ersten Herstellen einer Verbindung zum `altova.com` Master Licensing Server länger als 5 Tage (=120 Stunden) keine Verbindung mehr zum Master LicenseServer herstellen kann, lässt der Altova LicenseServer die Verwendung von Altova Software-Produkten, die mit dem Altova LicenseServer in Verbindung stehen, nicht mehr zu.

Wenn die Verbindung zum `altova.com` Master-Server verloren geht, wird dies auf dem Register "Messages" der Konfigurationsseite von LicenseServer protokolliert. Außerdem kann der Administrator den Altova LicenseServer so konfigurieren, dass automatisch eine entsprechende Benachrichtigung gesendet wird, wenn die Verbindung zu `altova.com` unterbrochen wird. Die Benachrichtigungseinstellungen werden auf der Seite "Configuration" auf dem Register "Settings" vorgenommen.

Aktivieren, Deaktivieren und Löschen einer Lizenz
Eine aktive Lizenz kann durch Auswahl der Lizenz und Klicken auf **Deactivate** deaktiviert werden. Eine inaktive Lizenz kann aktiviert (Schaltfläche **Activate**) oder gelöscht (Schaltfläche **Delete**) werden. Wenn eine Lizenz gelöscht wird, wird sie aus dem Lizenzpool entfernt. Eine gelöschte Lizenz kann durch Laden der Lizenzdatei, die sie enthält, wieder zum Lizenzpool hinzugefügt

 © 2014 Altova GmbH

werden. Wenn eine Lizenzdatei erneut hochgeladen wird, werden nur Lizenzen, die sich noch nicht im Lizenzpool befinden, zum Lizenzpool hinzugefügt; Lizenzen, die sich bereits im Pool befinden, werden nicht erneut hinzugefügt.

Symbole auf dem Register "License Pool"

Edit Assigned Licenses. Wird (in der Spalte "Assignments") zu jeder Lizenz angezeigt. Ruft das Dialogfeld Manage Licenses auf, in dem Sie dem Produkt neue Lizenzen zuweisen und zugewiesene Lizenzen ändern können..

Show License Information. Wird (in der Spalte "Assignments") zu jeder Lizenz angezeigt. Liefert Informationen zu den derzeit aktiven Clients.

Lizenzinformationen
Es werden die folgenden Informationen zur Lizenz angezeigt:

- *Status:* Es gibt die folgenden Werte: *Failed Verification* | *Activating* | *Active* | *Inactive* | *Blocked*. Siehe Lizenzstatus oben.
- *Name, Company:* Der Name und die Firma, die auf der Lizenz eingetragen sind. Diese Informationen wurden zum Zeitpunkt des Kaufs übermittelt.
- *Product, Edition, Version:* Die Version und Edition des lizenzierten Produkts.
- *Key, Expires in days, SMP (days left):* Der Lizenzschlüssel, mit dem das Produkt aktiviert wird und die Anzahl der verbleibenden Tage bis zum Ablauf der Lizenz. Jedes lizenzierte Produkt wird mit einem Support- und Wartungspaket (SMP), das eine bestimmte Anzahl von Tagen gültig ist, geliefert. In der Spalte *SMP* wird angegeben, wie lange das SMP noch gültig ist.
- *Users* | *CPU Cores:* Die Anzahl der Benutzer oder CPU-Kerne, für die die Lizenz ausgestellt wurde. Im Fall des Altova-Produkts MobileTogether Server werden die Lizenzen auf Basis der Anzahl der MobileTogether Clients, die mit MobileTogether Server verbunden werden, zugewiesen. Im Fall aller anderen Altova Server-Produkte werden die Lizenzen auf Basis der CPU-Kerne zugewiesen (*siehe Anmerkung unten*).
- *Assignments:* Zugriff auf die Bearbeitungsdialogfelder und die Informationen zu einzelnen Lizenzen.

Anmerkung zu Prozessorkernen und Lizenzen
Die Lizenzierung von Altova Server-Produkten basiert *mit Ausnahme von MobileTogether Server**** auf der Anzahl der auf dem Produktrechner verfügbaren Prozessorkerne. So hat z.B. ein Dual-Core-Prozessor zwei Prozessorkerne, ein Quad-Core-Prozessor hat vier Kerne, ein Hexa-Core-Prozessor hat sechs Kerne, usw. Die Anzahl der für ein Produkt auf einem bestimmten Server-Rechner lizenzierten Kerne muss größer oder gleich der Anzahl der auf diesem Server verfügbaren Kerne sein, unabhängig davon, ob es sich um einen physischen Rechner oder eine Virtual Machine handelt.

Wenn ein Server z.B. acht Kerne hat (ein Octa-Core-Prozessor), müssen Sie eine Lizenz für mindestens acht Kerne erwerben. Sie können Lizenzen auch kombinieren, um das Produkt für die entsprechende Anzahl von Kernen zu lizenzieren. So können z.B. anstelle einer Lizenz für 8 Kerne auch zwei Lizenzen für 4 Kerne für einen Octa-Core-Prozessor verwendet werden.

Wenn Sie einen Server-Rechner mit eine großen Anzahl von CPU-Kernen verwenden, aber nur geringe Datenmengen verarbeiten müssen, können Sie auch eine Virtual Machine erstellen, der eine geringere Anzahl an Kernen zugewiesen ist und eine Lizenz für diese Anzahl an Kernen erwerben. In diesem Fall ist die Verarbeitungsgeschwindigkeit natürlich geringer als bei Verwendung aller Kerne des Rechners.

Anmerkung: Jede Lizenz kann immer nur für einen Client gleichzeitig verwendet werden, selbst wenn die Lizenzkapazität noch nicht ausgeschöpft ist. Wenn z.B. eine 10-Kern-Lizenz für einen Client-Rechner mit 6 CPU-Kernen verwendet wird, so können die Lizenzen für die restlichen 4 Kerne nicht gleichzeitig für einen anderen Client-Rechner verwendet werden.

*** MobileTogether Server-Lizenzen werden auf Basis der Anzahl der Benutzer, d.h. auf Basis der Clients, die auf MobileTogether Server zugreifen, zugewiesen.

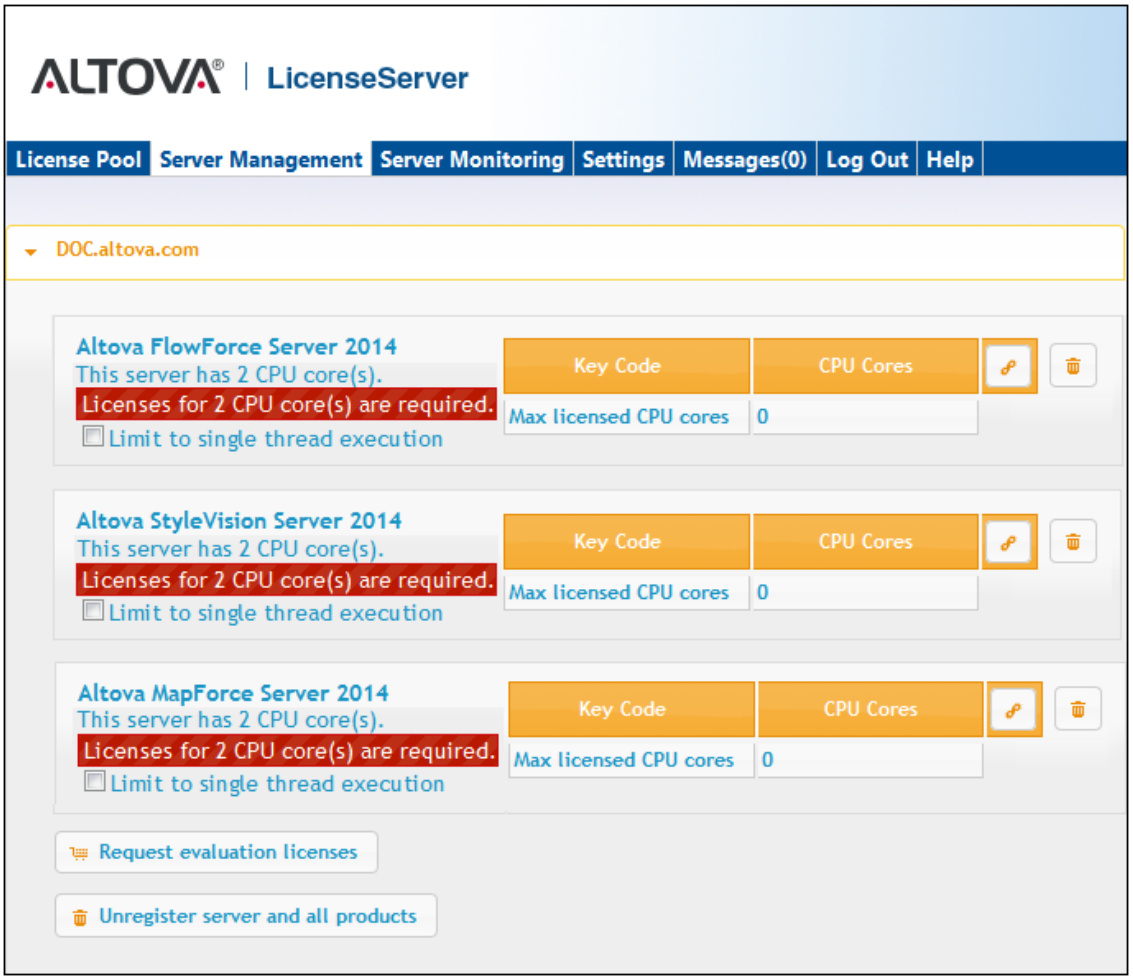
 © 2014 Altova GmbH

10.7.2 Server Management

In diesem Abschnitt werden folgende Schritte beschrieben:

- Symbole auf dem Register "Server Management"
- Zuweisen von Lizenzen
- Ein einziger Client, der unter verschiedenen Namen registriert ist
- Anfordern einer Testlizenz
- Aufheben der Registrierung für Produkte

Auf dem Register **Server Management** (*Abbildung unten*) können Sie registrierten Produkten Lizenzen zuweisen.



Beachten Sie zum Register "Server Management" die folgenden Punkte:

- Jedes Produkt wird unter dem Namen seines Client-Rechners aufgelistet. In der Abbildung oben sehen Sie einen Client-Rechner namens `Doc.altova.com`, auf dem drei Altova-Produkte installiert sind, die am LicenseServer registriert wurden. Wenn auf diesem LicenseServer ein Altova-Produkt auf einem anderen Client-Rechner registriert

wird, so wird auch dieser Client-Rechner mit seinen registrierten Produkten auf dem Register "Server Management" aufgelistet.

- Jedes registrierte Altova-Produkt auf einem Client-Rechner hat seinen eigenen *Keycode* Eintrag, der den Keycode einer Lizenz enthält. Um den Keycode eines registrierten Produkts zuzuweisen, klicken Sie auf die dazugehörige Schaltfläche **Edit Assigned Licenses** und wählen Sie die erforderliche Lizenz aus den im Lizenzpool für dieses Produkt (z.B. FlowForce Server) verfügbaren Lizenzen aus. Eine ausführlichere Beschreibung zu diesem Verfahren finden Sie weiter unten.
- Außerdem gibt es zu jedem Produkt (mit Ausnahme von MobileTogether Server) eine Zeile, in der angegeben wird, wie viele CPU-Kerne für dieses Produkt auf diesem Client lizenziert werden müssen. Wenn die Anzahl der lizenzierten Kerne geringer ist als die erforderliche Anzahl, werden diese Informationen rot markiert (*siehe Abbildung oben*). Die Anzahl der zu lizenzierenden CPU-Kerne entspricht der Anzahl der CPU-Kerne auf diesem Client. Diese Zahl wird von LicenseServer vom Client-Rechner abgerufen.

Single-Thread-Ausführung
Wenn im Lizenzpool eine Lizenz für nur einen Prozessorkern verfügbar ist, können Sie einem Rechner mit mehreren Kernen diese Lizenz für einen Kern zuweisen. In diesem Fall führt der Rechner das Produkt an einem einzigen Kern aus. Dadurch verlangsamt sich die Verarbeitungsgeschwindigkeit, da kein Multi-Threading (welches bei mehreren Prozessorkernen möglich wäre) zur Verfügung steht. Das Produkt wird auf diesem Rechner im Single Thread-Modus ausgeführt.

Um einem Mehrkernrechner eine Lizenz für nur einen Kern zuzuweisen, aktivieren Sie für das entsprechende Produkt das Kontrollkästchen *Limit to single thread execution*.

Symbole auf dem Register "Server Management"

Edit Assigned Licenses. (Zugewiesene Lizenzen bearbeiten) Steht für jedes Produkt zur Verfügung. Ruft das Dialogfeld "Manage Licenses" auf, in dem dem Produkt neue Lizenzen zugewiesen und bereits zugewiesene bearbeitet werden können.

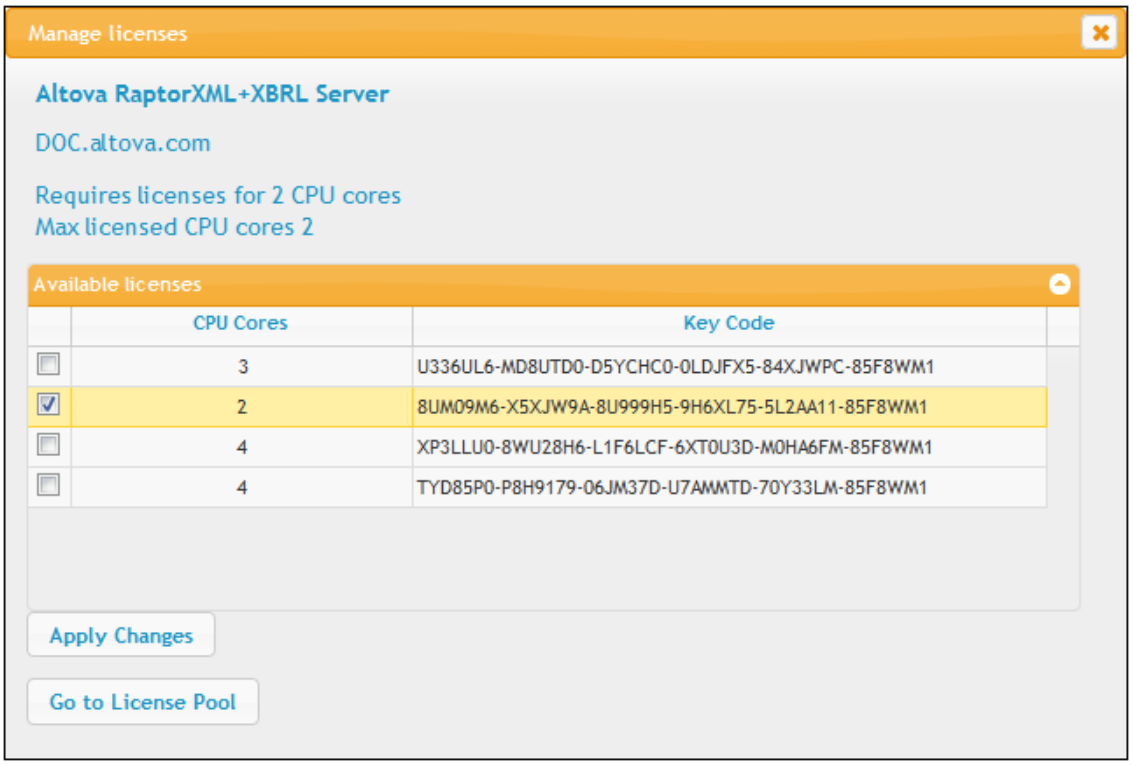
Show Licenses. (Lizenzen anzeigen) Wird zu jeder Lizenz angezeigt. Damit wechseln Sie zum Register "License Pool", wo die ausgewählte Lizenz markiert wird, damit Sie die Lizenzdetails lesen können.

Unregister This Product. (Registrierung für dieses Produkt aufheben) Steht für jedes Produkt zur Verfügung. Die Registrierung des ausgewählte Produkts (auf dem ausgewählten Client-Rechner) wird beim LicenseServer aufgehoben.

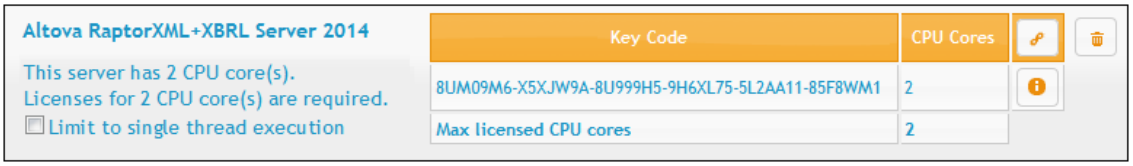
Zuweisen einer Lizenz

 © 2014 Altova GmbH

Um einem registrierten Produkt eine Lizenz zuzuweisen, klicken Sie auf die Schaltfläche **Edit Assigned Licenses** dieses Produkts. Daraufhin wird das Dialogfeld "Manage Licenses" (*Abbildung unten*) aufgerufen.



Klicken Sie nach Auswahl der Lizenz(en) auf **Apply Changes**. Daraufhin wird/werden die Lizenz(en) diesem Produkt zugewiesen und auf dem Register "Server Management" angezeigt (*siehe Abbildung unten*).



Ein einziger Client, der unter verschiedenen Namen registriert ist
Wenn ein Client-Rechner mehr als einmal auf dem LicenseServer registriert wurde, wird er unter Umständen auf dem Register "Server Management" in Form mehrerer Einträge unter mehreren Namen aufgelistet. Dies kann z.B. vorkommen, wenn ein Rechner mit dem Host-Namen in einer anderen Form erneut registriert wurde.

Um sicherzustellen, dass demselben unter unterschiedlichen Namen registrierten Rechner nicht mehrere Lizenzen zugewiesen werden, sollten Sie die Registrierung für überflüssige Client-Rechner-Einträge aufheben. Klicken Sie dazu für diese Rechner auf die Schaltfläche **Unregister server and all products**. *(Bitte beachten Sie: In dieser Dokumentation werden Rechner, die*

eigentlich Server ihrer eigenen Produkte sind, als Client-Rechner von LicenseServer behandelt). Außerdem kann es zu Lizenzkonflikten kommen, wenn dieselbe Lizenz demselben Rechner unter seinen verschiedenen Namen mehrmals zugewiesen ist. Um diese beiden Situationen zu vermeiden (überflüssige Lizenzen und mehrfache Zuweisungen einer einzigen Lizenz), wird empfohlen, überzählige Registrierungen eines einzigen Client-Rechners zu entfernen.

Im Folgenden sehen Sie, in welcher Form ein Rechnername auf dem Register "Server Management" aufgelistet werden kann:

- *Host-Name mit Domänenname (der vollqualifizierte Domänenname FQDN) wie z.B.: "win80-x64_1.my.domain.com" oder "Doc3.my.domain.com".* Der Name wird in dieser Form aufgeführt, wenn der Host-Name des Rechners (mit oder ohne die Domänendaten) als Argument des CLI-Befehls `licenseserver`, mit dem das Serverprodukt auf dem LicenseServer registriert wird, übergeben wird. Beispiel: `<AltovaServerProduct> licenseserver Doc3`. Dabei wird ein FQDN wie z.B. der folgende erzeugt: `Doc3.my.domain.com`.

  Ein FQDN wird auch erzeugt, wenn auf Windows 7-Systemen `localhost` als Host-Name angegeben wird..

- *Host-Name ohne Domänenname. Beispiel: "win80-x64_1" oder "Doc3".* In dieser Form wird der Name auf Windows 8-Systemen angeführt, wenn als Rechnername `localhost` verwendet wird..

- *localhost*. In manchen Fällen wird auch `localhost` als Rechnername angezeigt.

Anmerkung: Wenn ein Rechner bei Installation eines Altova-Server-Produkts auf Windows-Rechnern automatisch auf dem LicenseServer registriert wird, verwendet das Installationsprogramm `localhost` als Rechnernamen.

Anfordern einer Testlizenz
Sie können für jedes der auf dem LicenseServer registrierten Altova-Produkte eine kostenlose 30-Tage-Testlizenz anfordern. Klicken Sie dazu am unteren Rand des Registers "Server Management" auf die Schaltfläche **Request Evaluation Licenses**. Daraufhin wird ein Dialogfeld mit einer Liste der auf dem LicenseServer registrierten Altova Server-Produkte (auf diesem Client-Rechner) angezeigt. Versehen Sie die Produkte, für die Sie eine Testlizenz anfordern möchten, mit einem Häkchen, füllen Sie anschließend die Registrierungsfelder aus und senden Sie die Anfrage ab. Sie erhalten daraufhin eine E-Mail mit der/den 30-Tage-Testlizenzen von Altova. Die Anzahl der Prozessorkerne, für die die Lizenz pro Produkt gültig ist, entspricht genau der Anzahl der für das zum Zeitpunkt der Anforderung für das Produkt benötigten Anzahl. Speichern Sie die Lizenzen und laden Sie sie in den Lizenzpool.

Aufheben der Registrierung für Produkte
Jedes auf dem LicenseServer registrierte Altova-Produkt wird auf dem Register "Server Management" unter dem Namen seines Client-Rechners aufgelistet. Rechts daneben befindet sich ein **Unregister** Symbol. Klicken Sie darauf, um die Registrierung für das Produkt

aufzuheben. Wenn dem Produkt eine Lizenz zugewiesen wurde, wird die Zuweisung beendet, sobald die Registrierung für das Produkt aufgehoben wird. Um die Registrierung für alle Produkte aufzuheben, klicken Sie am unteren Rand des Registers "Server Management" auf die Schaltfläche **Unregister Server and All Products** (*siehe erste Abbildung in diesem Abschnitt)*.

Um ein Produkt erneut auf dem LicenseServer zu registrieren, gehen Sie zur Setup-Seite dieses Produkts oder rufen Sie sein CLI auf und registrieren Sie das Produkt. *Siehe*: Registrieren von FlowForce Server, Registrieren von MapForce Server, Registrieren von StyleVision Server und Registrieren von RaptorXML(+XBRL) Server.

Nähere Informationen finden Sie im Abschnitt Zuweisen von Lizenzen zu registrierten Produkten.

10.7.3 Server Monitoring

Das Register **Server Monitoring** enthält eine Übersicht über die Server, auf denen derzeit lizenzierte Altova Produkte ausgeführt werden. Das Register enthält Produktinformationen zusammen mit Informationen über Benutzer und Lizenzen.

© 2014 Altova GmbH

10.7.4 Settings

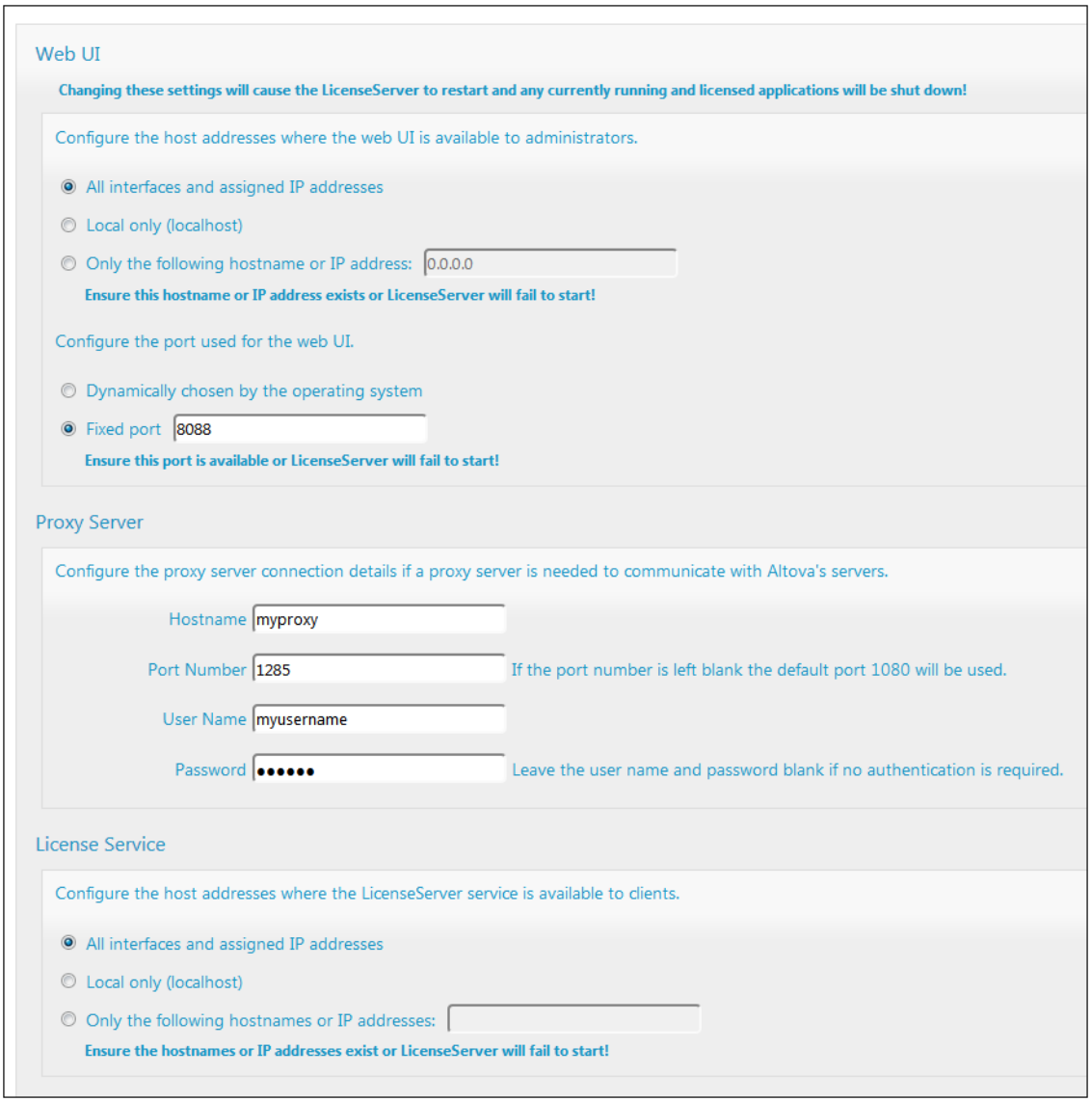
In diesem Abschnitt werden folgende Schritte beschrieben:

- Netzwerkeinstellungen
- Benachrichtigungseinstellungen
- Diverse Einstellungen

Im Folgenden finden Sie eine Beschreibung des Registers **Settings** (Einstellungen). Darauf können Sie die folgenden Einstellungen vornehmen:

- das Passwort, mit dem Sie sich auf dem LicenseServer einloggen. Geben Sie das gewünschte Passwort ein und klicken Sie auf **Change Password.**
- Testen Sie die Verbindung durch Klicken auf **Test Connection to Altova**. Beachten Sie, dass neue Einstellungen vor Prüfen der Verbindung (durch Klicken auf die Schaltfläche **Speichern**) gespeichert werden müssen. Die Schaltfläche **Test Connection to Altova** ist während des Überprüfungsvorgangs deaktiviert und wird erst aktiv, sobald die Überprüfung beendet worden ist.
- Client-Statistiken
- die Netzwerkeinstellungen für die webbasierte Konfigurationsseite (Web UI), der für die Internetverbindung verwendete Proxy-Server (falls verwendet) und die Einstellungen für LicenseServer (License Service). Diese Einstellungen sind weiter unten unter Netzwerkeinstellungen beschrieben.
- Die E-Mail-Server-Einstellungen und die E-Mail-Adresse, an die bei Bedarf LicenseServer-Benachrichtigungen gesendet werden sollen. Diese Einstellungen sind weiter unten unter Benachrichtigungseinstellungen beschrieben.
- Nachdem Sie eine Einstellung geändert haben, kicken Sie am unteren Rand der Seite auf **Save**. Geänderte Einstellungen werden erst nach dem Speichern aktiv.

Netzwerkeinstellungen
Ein Administrator kann Netzwerkadressen für die LicenseServer-Konfigurationsseite und LicenseServer definieren:


Web UI
Changing these settings will cause the LicenseServer to restart and any currently running and licensed applications will be shut down!
Configure the host addresses where the web UI is available to administrators.
All interfaces and assigned IP addresses
Local only (localhost)
Only the following hostname or IP address:
0.0.0.0
Ensure this hostname or IP address exists or LicenseServer will fail to start!
Configure the port used for the web UI.
Dynamically chosen by the operating system
Fixed port
8088
Ensure this port is available or LicenseServer will fail to start!
Proxy Server
Configure the proxy server connection details if a proxy server is needed to communicate with Altova's servers.
Hostname
myproxy
Port Number
1285
If the port number is left blank the default port 1080 will be used.
User Name
myusername
Password
Leave the user name and password blank if no authentication is required.
License Service
Configure the host addresses where the LicenseServer service is available to clients.
All interfaces and assigned IP addresses
Local only (localhost)
Only the following hostnames or IP addresses:
Ensure the hostnames or IP addresses exist or LicenseServer will fail to start!

- *Web UI:* Als zulässige IP-Adressen können alle Schnittstellen und IP-Adressen auf diesem Rechner oder nur eine fixe Adresse definiert werden. Die Ports können entweder dynamisch ermittelt werden oder fix sein. Dadurch können die verschiedensten IP-Adressen:Port-Einstellungen konfiguriert werden. Die Standardporteinstellung ist **8088**.
- *Proxy Server (verfügbar ab v1.3):* Wenn für die Internetverbindung ein Proxy-Server verwendet wird, müssen die Informationen des Proxy-Servers im Bereich "Proxy Server" (*siehe Abbildung oben*) eingegeben werden. Diese Felder müssen nur ausgefüllt werden, wenn ein Proxy-Server verwendet wird. Außerdem ist für Proxy-Server oft keine Authentifizierung (Benutzername und Passwort) erforderlich. In diesem Fall können diese beiden Felder leer bleiben. Um LicenseServer für den Proxy-Server zu konfigurieren, geben Sie den Host-Namen und, falls erforderlich, die Port-Nummer des Proxy-Servers ein.
- *License Service:* Als zulässige IP-Adressen können alle Schnittstellen und IP-Adressen auf diesem Rechner oder eine fixe Adresse definiert werden. Verwenden Sie bei Auflistung von Hostnamen und/oder IP-Adressen eine kommagetrennte Liste ohne Leerzeichen (z.B.: `hostname1,IPAddress1,hostname2`). Die Port-Nummer ist mit **35355**

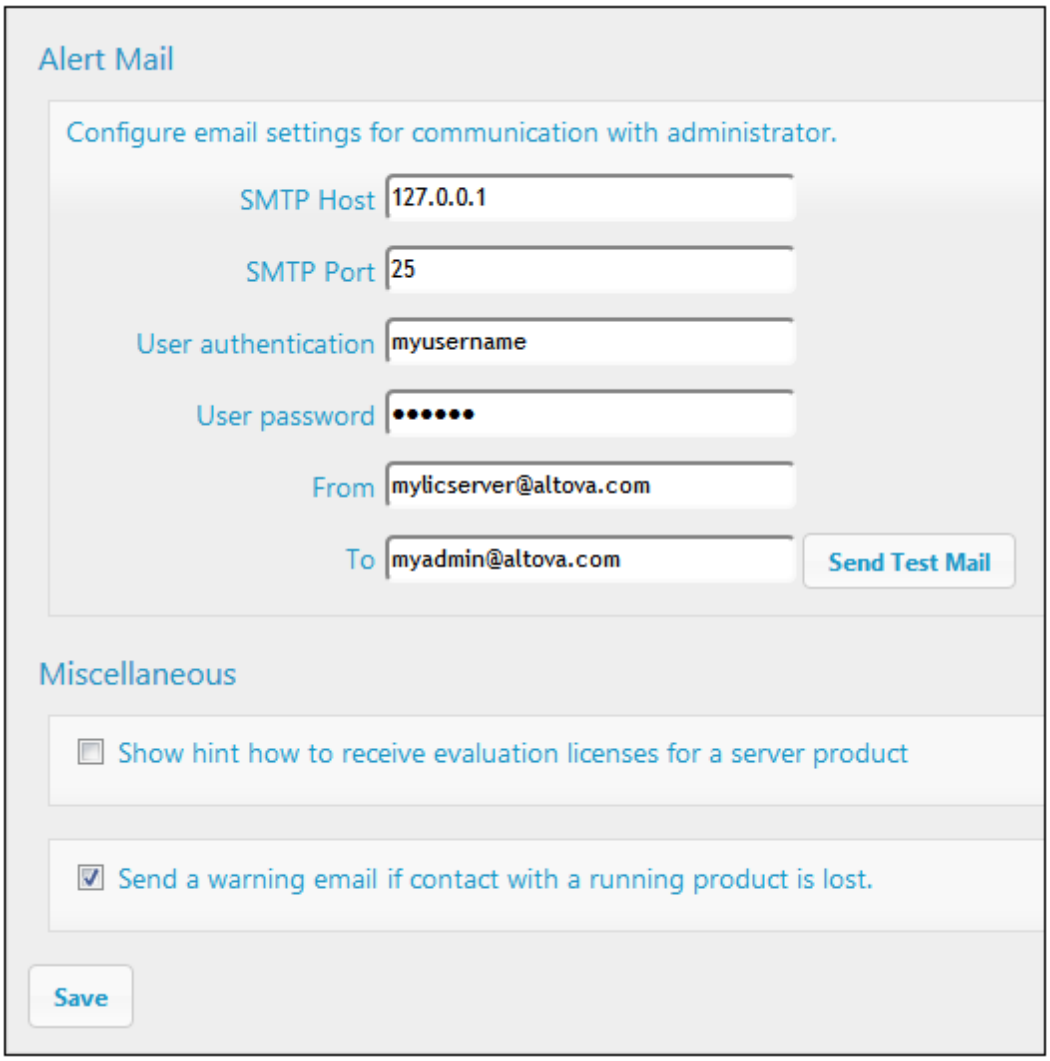
fixiert.

Standardmäßig erlauben diese Einstellungen den unbeschränkten Zugriff auf LicenseServer und seine Konfigurationsseite über das Netzwerk, mit dem LicenseServer verbunden ist. Wenn Sie den Zugriff auf LicenseServer oder seine Konfigurationsseite einschränken möchten, geben Sie die entsprechenden Einstellungen ein und klicken Sie auf **Save**.

Führen Sie einen Verbindungstest durch (*siehe oben*), um zu überprüfen, ob die Einstellungen korrekt sind.

Benachrichtigungseinstellungen
Der Altova License Server muss eine aufrechte Verbindung zum `altova.com` Server haben. Wenn die Verbindung länger als 24*5 Stunden (5 Tage) unterbrochen ist, vergibt der LicenseServer keine Lizenzen mehr, sodass Arbeitssitzungen mit Altova-Produkten, die bei LicenseServer registriert sind, unterbrochen werden könnten.

Wenn die Verbindung unterbrochen wird, kann eine Warnungsnachricht an eine E-Mail-Adresse gesendet werden. Die Einstellungen, um Benachrichtigungen an die E-Mail-Adresse des Administrators zu senden, werden im Bereich "Alert Mail" (*siehe Abbildung unten*) vorgenommen.



SMTP Host und *SMTP Port* sind die Zugriffseinstellungen des E-Mail-Servers, von dem aus die E-Mail gesendet wird. *User Authentication* und *User Password* sind die Anmeldeinformationen des Benutzers, mit denen er auf den E-Mail-Server zugreift. In das Feld *From* wird die E-Mail-Adresse eingetragen, von der aus die E-Mail gesendet wird. In das Feld *To* wird die E-Mail-Adresse des Empfängers eingetragen.

Klicken Sie, wenn Sie mit dem Konfigurieren fertig sind, auf **Save**. Nachdem Sie die Alert Mail-Einstellungen gespeichert haben, werden bei jedem wichtigen Ereignis, wie z.B. wenn eine Verbindung zu `altova.com` unterbrochen ist, E-Mail-Benachrichtigungen an den Administrator gesendet. Diese Ereignisse werden auch auf dem Register "Messages" protokolliert, wo Sie das Log ansehen können.

Diverse Einstellungen

Show hints for receiving and deploying evaluation licenses
Wenn Sie dieses Kontrollkästchen (*siehe Abbildung oben*) aktivieren, wird im oberen Bereich der

 © 2014 Altova GmbH

Konfigurationsseite eine kurze Anleitung zum Testen und Bereitstellen von Evaluierungslizenzen angezeigt.

Send a warning email if contact with a running product is lost
Wenn die Verbindung zu einem lizenzierten laufenden Produkt unterbrochen wird, wird von der *Von*-Adresse an die *An*-Adresse eine Warungsmeldung gesendet.

10.7.5 Messages, Log Out

Auf dem Register **Messages** werden alle Meldungen, die die Lizenzen im Lizenzpool des LicenseServers betreffen, angezeigt. Zu jeder Meldung gibt es eine **Löschen**-Schaltfläche, über die Sie diese Meldung löschen können.

Das Register **Log Out** dient als Schaltfläche zum Ausloggen. Wenn Sie darauf klicken, werden Sie sofort ausgeloggt. Anschließend wird die Login-Maske angezeigt.

© 2014 Altova GmbH

Index

.

.NET Erweiterungsfunktionen,
Datentypkonvertierungen, .NET in XPath/XQuery, 431
Datentypkonvertierungen, XPath/XQuery in .NET, 430
für XSLT und XQuery, 425
Instanzmethoden, Instanzfelder, 429
Konstruktoren, 427
statische Methoden, statische Felder, 428
Übersicht, 425
.NET-Schnittstelle, 5

A

Altova extensions,
chart functions (see chart functions), 364
Altova LicenseServer,
(siehe LicenseServer), 436
Altova ServiceController, 443

B

Befehlszeile,
Optionen, 117
und XQuery, 96
Zusammenfassung über die Verwendung, 46

C

COM-Schnittstelle, 5

D

Diagrammfunktionen,
Beispiel, 407
Diagrammdatenstruktur, 402
Liste, 398

E

Einrichten, 12
auf Mac OS X, 27
unter Linux, 21
unter Windows, 13
E-Mail-Benachrichtigungen, 489
Erweiterungsfunktionen für XSLT und XQuery, 414
Erweiterungsfunktionen in .NET für XSLT und XQuery,
siehe .NET Erweiterungsfunktionen, 425
Erweiterungsfunktionen in Java für XSLT und XQuery,
siehe Java-Erweiterungsfunktionen, 415
Erweiterungsfunktionen in MSXSL Scripts, 432

F

FlowForce Server,
am License Server registrieren, 460

G

Global resources, 41

H

Hilfebefehl auf CLI, 111
HTTP-Schnittstelle, 5, 130
Client Request, 141
Einrichten des Servers, 132
Serverkonfiguration, 136
Sicherheitsfragen, 43

I

Informationen zum Netzwerk, 438
Insrtallation unter Windows, 14
Installation auf Mac OS X, 28

© 2014 Altova GmbH

Installation unter Linux, 22

J

Java Erweiterungsfunktionen,
Datentypkonvertierungen, Java in Xpath/XQuery, 423
Instanzmethoden, Instanzfelder, 421
Konstruktoren, 420
statische Methoden, statische Felder, 421
Java extension functions,
datatype conversions, XPath/XQuery to Java, 422
Java-Erweiterungsfunktionen,
benutzerdefinierte JAR-Dateien, 419
benutzerdefinierte Klassendateien, 416
für XSLT und XQuery, 415
Übersicht, 415
Java-Schnittstelle, 5

K

Kataloge, 33
Konfigurationsseite, 477
auf Mac OS X öffnen, 453
unter Linux öffnen, 450
unter Windows öffnen, 447
URL, 447
URL (Linux), 450
URL (Mac OS X), 453

L

License Pool, 478
LicenseServer,
Einstellungen, 489
FlowForce Server registrieren, 460
Installation auf Mac OS X, 442
Installation unter Linux, 440
Installation unter Windows, 439
Konfigurationsseite, 477
MapForce Server registrieren, 465
Schnittstelle zu, 477
Schritte beim Zuweisen von Lizenzen, 444
starten, 445
StyleVision Server registrieren, 466
LicenseServer-Konfigurationsseite,
(siehe Konfigurationsseite), 447, 450, 453
Linux,
Installation unter, 22
Lizenzbefehle auf CLI, 113
Lizenzen,
laden, 456, 478
zuweisen, 471, 483
Lizenzen laden, 456, 478
Lizenzen zuweisen, 471
Lizenzierung auf Mac OS X, 31
Lizenzierung unter Windows, 17, 25
Lizenzpool, 456
Logout, 494
Lokalisierung, 114

M

Mac OS X,
Installation, 28
Lizenzierung, 31
MapForce Server,
am License Server registrieren, 465
Messages, 494
msxsl:Script, 432

N

Netzwerkeinstellungen, 489

P

Passwort,
Standardpasswort beim Start, 447
Python,
Sicherheitsfragen, 43
Python API,
XML API, 179
XSD API, 194
Python-API,
Auftragsobjekt, 178
Python-Beispielskript, 166, 171

© 2014 Altova GmbH

Python-Schnittstelle, 5, 160
Skripts ausführen, 165
Skripts erstellen, 162

R

RaptorXML,
Befehlszeilenschnittstelle, 5
Editionen und Schnittstellen, 5
Einführung, 3
Funktionalitäten, 8
HTTP-Schnittstelle, 5
Python-Schnittstelle, 5
Schnittstellen zu COM, Java, .NET, 5
Systemanforderungen, 7
unterstützte Spezifikationen, 10
Register "Server Management", 471
Registrieren von FlowForce Server am LIcenseServer, 460
Registrieren von MapForce Server am LIcenseServer, 465
Registrieren von StyleVision Server am LIcenseServer, 466

S

Schnittstellen,
Übersicht, 5
Scripts in XSLT/XQuery,
siehe Erweiterungsfunktionen, 414
Server Management (Register), 483
Server Monitoring (Register), 488
Serverkonfiguration, 136
ServiceController, 443
Settings, 489
Sicherheitsfragen, 43
Standardpasswort, 447
StyleVision Server,
am License Server registrieren, 466

V

Validierung,
DTD, 59
von beliebigen Dokumenten, 67
von XML-Instanzen anhand einer DTD, 49
von XML-Instanzen anhand einer XSD, 53
von XQuery-Dokument, 102
XSD, 62
XSLT-Dokument, 90
Verwaltungsschnittstelle, 477

W

Windows,
Installation, 14
Lizenzieren unter, 17, 25
Wohlgeformtheitsprüfung, 71

X

XML-Kataloge, 33
XQuery,
Erweiterungsfunktionen, 414
XQuery-Ausführung, 97
XQuery-Befehle, 96
XQuery-Dokumentvalidierung, 102
XSLT,
Erweiterungsfunktionen, 414
XSLT Dokument,
validieren, 90
XSLT-Befehle, 82
XSLT-Transformation, 83

Z

Zuweisen von Lizenzen, 483

© 2014 Altova GmbH